Rheinische Friedrich-Wilhelms-Universität Bonn
Institute of Computer Science
Master Thesis Computer Science

Analyzing and Predicting Simulation Results using Oriented Bounding Boxes and Recurrent Neural Networks

Sara Vera Hahner

Born 7th January 1995 in Fulda, Germany

6th December 2019

Advisor: Prof. Dr. Reinhard Klein

Institute of Computer Science II

Second Advisor: Prof. Dr. Jochen Garcke

Institute for Numerical Simulation

Fraunhofer Institute for Scientific Computing and Algorithms SCAI

Contents

Notation

$\mathbb{N}$	Set of natural numbers: $\{0, 1, 2, 3, \dots\}$
$\mathbb{R}$	Set of real numbers
I_p	$p \times p$ identity matrix
$\mathcal{X}$	point cloud $\mathcal{X} = \{X_i \mid i = 1, ..., N\} \subset \mathbb{R}^d$
$\mathcal{CH}(\mathcal{X})$	Convex Hull of point set $\mathcal{X}$ defined by N_C vertices
$SO(d, \mathbb{R})$	$= \{R \in \mathbb{R}^{d \times d} \mid R^T R = I_n = RR^T, det(R) = 1\}$, all the orthogonal and real $d \times d$-matrices in $\mathbb{R}^{d \times d}$
$\mathbf{X}$	an OBB's center in $\mathbb{R}^d$
R	an OBB's rotation in $SO(d, \mathbb{R})$
$\mathbf{\Delta}$	an OBB's extensions in $\mathbb{R}^d$
$BB_{opt}(\mathcal{X})$	optimal oriented bounding box of $\mathcal{X}$
$BB_{PCA}(\mathcal{X})$	PCA-based bounding box of $\mathcal{X}$
$BB_{HYBBRID}(\mathcal{X})$	Bounding box of $\mathcal{X}$ obtained by HYBBRID algorithm
$\kappa_{d,i}$	General approximation factor of BB_{PCA} in d dimensions using i-dimensional faces to define the convex hull
$\mathcal{A}$	$\mathcal{A} = \{\mathcal{A}_i, i = 1, \dots, M\}$ the population for the genetic algorithm
P	Set of carparts
SIM	Set of simulations
(p, sim)	Tupel referencing the part $p \in P$ and simulation $sim \in SIM$
S	Sequence of length T_{total}
S_{in}	First T_{in} timesteps of S, inputted to LSTM Autoencoder
S_{pred}	Last $T_{total} - T_{in}$ timesteps of S
$\mathbf{h}$	Hidden Representation of S_{in} created by LSTM Autoencoder in $\mathbb{R}^l$
$\mathbf{h}_{emb}$	2D-embeddings of hidden representation $\mathbf{h}$

Abbreviations

OBB	Oriented Bounding Box
AABB	Axis Aligned Bounding Box
PCA	Principal component analysis
HYBBRID	Hybrid Bounding Box Rotation Identification
t-SNE	t-distributed stochastic neighbor embedding
RNN	Recurrent neural network
LSTM	Long short-term memory
MSE	Mean squared error
SVM	Support vector machine
STD	Standard Deviation

1. Introduction

Computer-aided simulations play a vital role in the development of products, as they enable simpler, faster and more cost-effective investigations of systems. An important example is car crash analysis, in which one studies the influence of model parameters on personal safety, plastic deformation, strain or rupture. The tests are simulated to a high degree of detail at high computational cost for different parameter settings, which are inputs to the simulations. Naturally, exhaustive evaluation of the parameter space to represent the full range of material and design choices is intractable.

For each simulation, a fine grid of points is placed over the component, the so called 3D surface mesh, and the displacements of the grid points are approximately computed on the basis of physical relationships. The results of a simulation are the positions of the individual grid points at different times, and in addition point-wise values of physical quantities. Engineers are especially interested in the deformation of specific car components. While a large variety of deformation processes is observed per car part, their result occurs in a small number of patterns, called modes of deformation, typically encountered in automotive engineering. In figure 1.1 the two distinct modes of deformation of the left front beams can be observed for the studied data set.

While deformation modes are dependent on model parameters, their behavior is accessible, albeit tediously, upon manifestation for engineers with access to the completed simulation run. In practice this detection of deformation modes is based on error-prone supervision of a non-trivial, hand-selected subset of nodes in critical car components.

In general, simulation results contain abundant features and are therefore high dimensional and unwieldy, which makes comparative analysis difficult. Intuitively, the trajectory of different grid points is highly correlated, not only in space but also over time. This redundancy in the data invites further analysis via feature learning reducing unimportant information and thus dimensionality. The low dimensional features capture distinguishing characteristics of the deformations, enabling clustering of simulation results into modes.

As soon as the deformation of a component is attributed to a deformation mode, there is little variance regarding its further course in the simulation. This motivates the in-situ detection, that is during the crash simulation, of deformation modes; we can then interrupt the simulation and anticipate targeted changes to the model parameters.

For the in-situ analysis of car crash simulations, I propose the application of an LSTM autoencoder trained on a shape representation by oriented bounding boxes instead of directly on the 3D surface meshes. The LSTM autoencoder is specialized on handling time sequences and can both reconstruct the input and predict the future timesteps of a sequence. It creates an additional low dimensional hidden representation which after successful training represents the features of the input time sequences and

allows its reconstruction and prediction. In this case the features encode the distinctive characteristics of the component's deformation.

Related Work

The analysis of car crash simulations has been studied with different machine learning techniques. For example, [BGG16; Boh+13; Zha+10; GS14] base their analysis on the finished simulation runs, whereas I focus on in-situ analysis of car crash simulations.

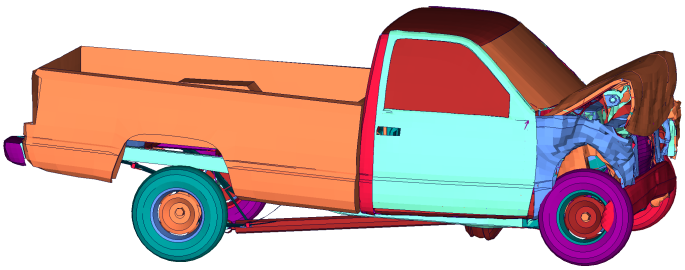
Other recent works study analysis and prediction of car crash simulation data under different problem definitions than in this work. The authors of [Gue+18] constructed a model that gives an estimation of the high-fidelity result for a new set of parameters without using the solver. The authors of [SSW17] applied Deep Learning on complete simulation data to detect anomalies in the results, which means they tested the structural mechanics for plausibility.

There are alternatives to oriented bounding boxes as low dimensional shape representations for car crash simulations. The authors of [IG19] construct low-dimensional shape representations via a projection of the data onto an eigenbasis stemming from a discrete Laplace Beltrami Operator. The data is then represented by the spectral coefficients. I want to point out the in-situ analysis of crash simluations presented in [AIG19], where the LSTM autoencoder from [SMS15] is applied to the spectral coefficients. The authors preselect a small subset of parts which are analyzed together, so that the model does not allow a part-wise study of deformation modes.

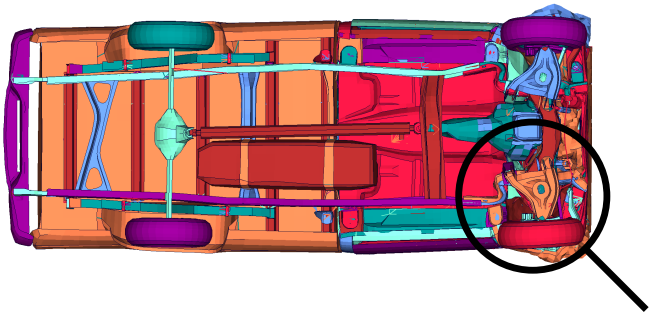
Using bounding boxes, changes to geometry and resolution are handled more readily than using spectral coefficients. In contrast to [AIG19], the neural network is applied part-wise and a lower dimensional shape representation is selected. Therefore, the model's size is smaller and the number of analyzed parts not limited anymore.

The majority of works about neural networks for geometries extend convolutional neural networks on non-euclidean shapes, including graphs and manifolds. [Mas+15; Mon+17; Bro+17; Lit+17] present some powerful extensions of neural networks for description learning and shape correspondence via the computation of local patches. Nevertheless, the networks are applied to the 3D-meshes directly and computationally expensive, when applied for every timestep of the high dimensional crash simulation.

Based on the shape deformation representation by [Gao+19], recent works studied generative modeling of meshes for 3D animation squences. [Qia+18] presented a bidirectional LSTM network that synthesizes and predicts 3D mesh animation sequences. They reduce the dimensionality of the shape deformation representation with a convolutional layer and utilize subsequent LSTM-layers to detect time dependencies. The authors of [Tan+18b; Tan+18a] presented variational autoencoders for deformation localization or shape generation of 3D mesh animation sequences. The shape representation yields good results, however, it solves an optimization problem at each vertex and requires identical connectivity over time. The architectures are tested on models of humans with considerably fewer nodes, which is why computational issues seem likely for the vertex-wise optimization problem.



(a) Side view of the car model.



(b) Bottom view of the car model.

(c) Modes of deformation of left front beams.

Figure 1.1.: Visualization of a selected crash simulation used for data collection after half of the simulation time. (a) Side view, (b) bottom view, (c) buckling behavior of left front beams (highlighted in bottom view).

Problem Definition

In the case of unsupervised feature learning with autoencoders, the distinctive patterns of the deformations are emphasized. The deformation of an uncompleted simulation run should be represented in one low dimensional feature vector per part, which focuses on the differences characterizing the deformation modes.

Instead of training the low dimensional features using 3D surface meshes, we utilize the simple but discriminate shape representation by oriented bounding boxes as an intermediate step, which are independent of resolution or geometry changes of the parts. In the long run, a simple representation as oriented bounding boxes might allow comparison of different car models with each other and use the knowledge from former simulation runs.

Changes in the model are transcribed by engineers into complex geometry changes, for example notching in specific positions or changes in the welds. This makes them difficult to encode as suitable variables for neural networks. Hence, the architecture analyzes deformation modes and concentrates on the understanding of the deformation behavior.

The procedure consists, therefore, of three steps, shown in figure 1.2: After preprocessing the car parts into oriented bounding boxes, an LSTM autoencoder calculates feature representations based on the deformation in the first timesteps of the simulation run. Autoencoders are powerful tools for unsupervised learning of representative features. We adapt a specialized LSTM autoencoder for handling timeseries, as described in [SMS15] for video sequences, that takes advantage of the temporal correspondence in the data. Additionally it predicts the future timesteps of the time sequences. Based on these representations, modes in the parts' deformation are detected and localized in the car using clustering and visualizations.

The research goals of this thesis are summarized to the following.:

- in-situ detection of deformation modes in a car model
- assignment to deformation patterns for effected parts as soon as possible during a simulation run
- in-situ prediction of car component positions at future timesteps

Contribution

- As part of the preprocessing process implementation and comparison of various algorithms for the approximation of oriented bounding boxes.

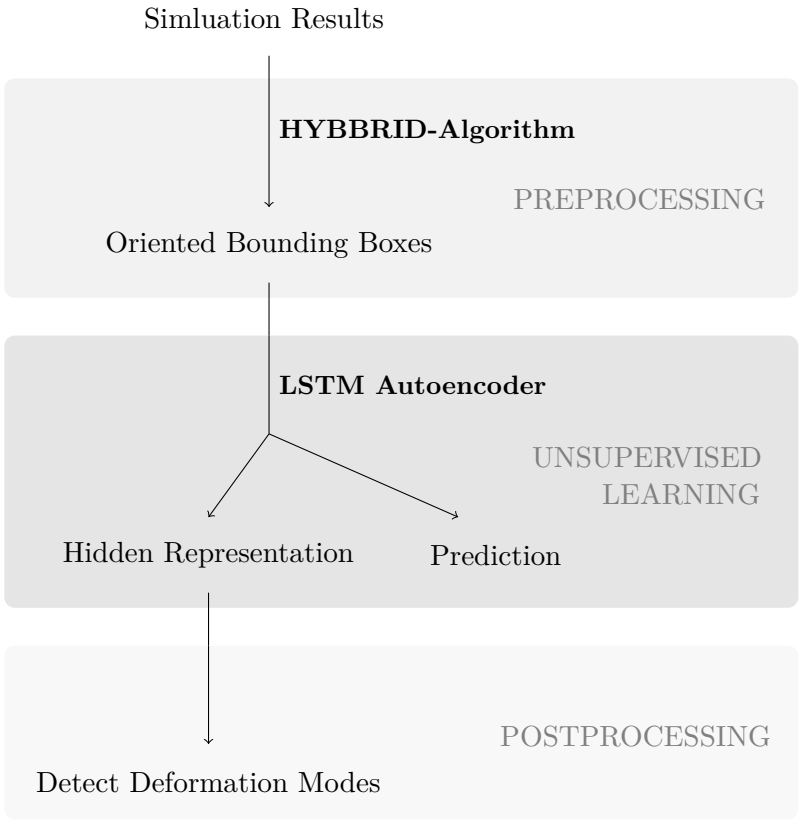


Figure 1.2.: *Input Data*: Simulation results, chapter 2.
Preprocessing: Oriented bounding boxes, chapter 3.
Unsupervised Learner: LSTM autoencoder creating feature representations summing up the whole sequence/deformation process, chapter 4.
Postprocessing: Detection of deformation modes using clustering and visualizations, chapter 4.

- Development of data specific improvements for the selected HYBBRID algorithm [CGM11] leading to faster and better results
- Application of the LSTM autoencoder [SMS15] to time sequences of oriented bounding boxes
  - Modification of the architecture for part-wise analysis of the input
  - Evaluation of the standard and composite model, which also allows prediction of time series
  - Extensive optimization of hyper parameters
- Definition of an SVM-based score to measure the quality of the deformation mode detection on the embedded low dimensional representations
- Evaluation of different versions of bounding boxes and their performance for deformation mode detection

Structure of the Thesis

In chapter 2, crash simulations are motivated along with points of interest for their analysis. Besides an explanation of the data set's structure, we discuss analysis methods and limitations when working with simulation data. Additionally, deformation modes in the buckling behavior of car components are defined.

The following chapter explains the preprocessing of the data by oriented bounding boxes. We discuss different algorithms to calculate or approximate the bounding boxes. Emerging weaknesses in the algorithms' quality and runtime are explained and amended in an application-specific manner, so that the approximation results improve and stabilize.

Chapter 4 introduces the model architecture, an LSTM autoencoder, starting with general information about autoencoders and long short term memory (LSTM). In addition, the postprocessing with t-SNE is explained. Figure 1.2 gives an overview of the structure of preprocessing, modelling and postprocessing.

Chapter 5 specifies the final training parameters, how the limitations on the number of parts can be circumvented and explains the quality measures applied to the predictions and low-dimensional features. Finally, in chapter 6 the results of the different model versions are compared. As a baseline, a direct prediction of the simulation results by nearest neighbors is provided.

The concluding chapter summarizes the results and gives an outlook on the possible applications and further developments of the investigated algorithm.

The proof of bounds on the volume of PCA-based bounding boxes and explanatory illustrations of the bounding boxes and deformation modes can be found in the appendix. This is followed by the bibliography.

Acknowledgements

At this point I would like to express my gratitude for the support I have received from all sides: First of all I would like to thank Prof. Dr. Klein for supervising my thesis as

well as the numerous and helpful meetings. I would like to thank Prof. Dr. Garcke for being the second advisor of the Master's thesis; I always had the opportunity to clarify ambiguities and to obtain constructive criticism. My special thanks go to Dr. Rodrigo Iza-Teran for his always outstanding support at Fraunhofer SCAI during my time at the institute and Amin Abbasloo for the many helpful suggestions to the topic. My thanks also go to Jacob, Akshat, Yannick and especially Jan for their attentive suggestions. I would also like to thank Bruno for his strong emotional support and my parents, who always have an open ear for my concerns and whose help I can count on at all times.

2. Simulation Data

Car models for crash simulation are generally described by a 3D surface mesh, which is structured into structural components of the car. In each simulation model parameters are varied, such as material characteristics, the geometry of car parts or their connections. The results of the crash simulations are analyzed for their plastic deformation, strain or rupture, to secure the passenger's safety.

The analysis of car crash simulations is challenging because of their high dimensionality and possible mesh-to-mesh correspondence problems between different simulations. Finally, the concept of deformation modes is explained.

2.1. Car Crash Simulation

The goal during the design of a car model is to secure a passenger's safety. The vehicle's structure should deform plastically in a way, that the occupants have sufficient survival space. The ability of cars to protect its occupants during an impact is called crashworthiness [Boi+04].

To improve a car's crashworthiness, engineers simulate car crashes for variations in geometry and material. Therefore, without the costly and time-consuming construction of new prototypes, changes in crash behavior can be analyzed in the early stages of development. Physical, destructive tests are usually only carried out for verification purposes to fulfill international standards and specific european regulations for passenger and vehicle safety [Eur18].

Given a computer model of the car, a finite element mesh out of finely sampled nodes is created for discretization for each car part. This 3D surface mesh is divided into triangles or quads for computing purposes. The material data is assigned to the nodes of the mesh and mathematical models of plastic deformations and other physical and mechanical effects are applied. Then boundary conditions and forces acting in the system are determined and an equation system is created. Finally, the nonlinear equation system is solved numerically (generally by commercial solvers on computing clusters) and the position of the surface mesh over time is output possibly along with additional information, such as acceleration or stress measures [Mey07]. The simulation results contain detailed information for hundreds of timesteps and more than three million nodes. The calculation of one simulation is costly in time, that means for one car model there are generally not more than 500 different simulation runs, and the results are very high dimensional.

2.2. Points of Interest

In automotive engineering the engineer distinguishes between various crash load cases, the most common being front, side and rear car crashes [Boi+04]. In this work the analysis concentrates on a data set of frontal-impact tests, wherein a vehicle impacts a solid concrete wall at a specified velocity.

For vehicle safety during frontal crashes the deformation of the crumple zone, which is the part of the car in front of the firewall separating passenger cabin and engine compartment, is crucial. The crumple zone should absorb the kinetic energy of the crash, plastically deform and avoid intrusion into the passenger compartment. The crumple zone's deformation is led by the deformation of the front beams and material changes of those components generally have a high impact on the result. Therefore, an engineer has high interest in understanding the beam's deformation during the analysis [Boi+04]. Figure 1.1 shows a snapshot of a selected crash simulation and highlights the front beam's deformation in different modes.

The overall goal in the design process is to thoroughly understand the dependencies between the model parameters and the deformation of the car parts and the parts' interaction. However, the deformation is dependent on the model parameters and, additionally, the model changes are often described in words, which is why dependencies cannot always be defined mathematically.

An intermediate goal is to understand and analyze the deformation of the car components. As stated in the introduction, the deformation behavior can mostly be clustered into different patterns, which will be called deformation modes, see figure 1.1c. If the deformation behavior of a car part shows exactly two different modes, this effect is called a bifurcation.

The assignment of a deformation to a pattern is mostly sufficient to decide whether it is satisfactory or not, because inside one cluster, there is generally only little variation, whereas the characteristics of two clusters differ strongly. The deformation modes should therefore be detected as soon as possible, optimally upon manifestation, which we study in the following chapters.

2.2.1. Challenges in the Analysis

In the analysis of modes in the deformation during car crash simulations different limitations emerge.

In the optimal case, the whole car would be analyzed at the same time. The size of the car models make this approach computationally expensive. A frequent solution is to study only one part or a small subset of parts that are preselected by relevance in previous simulations, for example [Boh+13] and [AIG19]. The choice of subsets of parts may be more efficient, but doesn't allow detection of interaction between parts. However, many deformations are induced by part interaction.

Additionally, the high computational cost of car crash simulations limits the available amount of training and test samples.

Finally, different mesh resolutions or changes in the geometry of two different models lead to a correspondence problem. The 3D surface meshes cannot be compared point-wise anymore requiring the use of shape representations of the meshes.

2.3. The TRUCK Dataset

As a simple car model, on which the proposed analysis method are tested, a Chevrolet C2500 pick-up truck from the National Crash Analysis Center[1] has been selected. The data set consists of $n_{simulations} = 196$ completed simulations[2] of a front crash (see figure 1.1), using the same truck, but with different material characteristics, which is a similar setup to [Boh+13]. For 9 car parts (the front and side beams and the bumper parts) the sheet thickness has been varied while keeping the geometry unchanged. Since there are no changes in the geometry, the one to one part-correspondence between different simulations is known and every part is indexed.

The results of the simulations consist of the three-dimensional geometry of the mesh for each timestep t, car part p and simulation sim. The set of car parts is denoted as P and the set of simulations as SIM. For the analysis we use the coordinates of the d_p nodes of the mesh $\mathcal{X}_1^{(p,sim)}, \mathcal{X}_2^{(p,sim)}, \ldots$ in $\mathbb{R}^{d_p \times 3}$, which are organized part-wise and move over time.

Every simulation has 152 saved timesteps, of which every fifth is taken for the analysis, and the model consists of approximately 290 car parts and 60,000 nodes. Hence, for each simulation there are $152 \cdot 60,000 \cdot 3$, approximately 27 million, output values.

For the chosen example a bifurcation in the left front beam's deformation is known beforehand [Boh+13] and illustrated in figure A.2 in the appendix. The deformation splits up into two different modes after approximately 50 timesteps. For 67 simulations the beam's deformation can be assigned to the first branch, the remaining 129 to the second branch.

[1] from NCAC `http://web.archive.org/web/*/www.ncac.gwu.edu/vml/models.html`
[2] computed with LS-DYNa `http://www.lstc.com/products/ls-dyna`

3. Preprocessing: Oriented Bounding Boxes

As outlined, an analysis of the deformation of all the car parts is challenging because of the size of the models, few simulation samples and the correspondence problem which occurs due to different mesh resolutions.

However, semantic design rules apply for the car model's components as is the case for most man-made models. Car wheels, the seats and beams are all of similar shape for different car models, which can be inferred once the general form and dimension are known. Because of these semantic rules, the model's parts can be approximated by their minimum bounding boxes which capture the part's translation, rotation and scale.

This chapter summarizes some theoretical statements of oriented bounding boxes in two and three dimensions and evaluates different algorithms with respect to runtime and quality. Finally, the chosen HYBBRID algorithm is customized to fit the characteristics of simulation data taking advantage of the correlation between the shapes for consecutive time steps.

3.1. Theory

A bounding box can be oriented (minimal) or axis aligned while enclosing a set of given points. Axis Aligned Bounding Boxes can be calculated in linear time by calculating the minimum and maximum value for all dimensions, but they don't perceive the rotation of an object.

Definition 1 (Axis Aligned Bounding Box)**.** *Given a finite set of N points $\mathcal{X} = \{X_i \mid i = 1, ..., N\} \subset \mathbb{R}^d$, the* axis aligned bounding box *(AABB) is defined as the cuboid, or rectangular parallelepiped, of minimal volume enclosing $\mathcal{X}$, whose edges are parallel to the coordinates systems' axes.*

An oriented bounding box is the bounding box with minimal volume enclosing the set of points.

Definition 2 (Oriented Bounding Box)**.** *Given a finite set of N points $\mathcal{X} = \{X_i \mid i = 1, ..., N\} \subset \mathbb{R}^d$, the* optimal oriented bounding box *(OBB) is defined as the arbitrarily oriented cuboid, or rectangular parallelepiped, of minimal volume enclosing $\mathcal{X}$.*

Each OBB is uniquely defined by

- its center $\mathbf{X} \in \mathbb{R}^d$

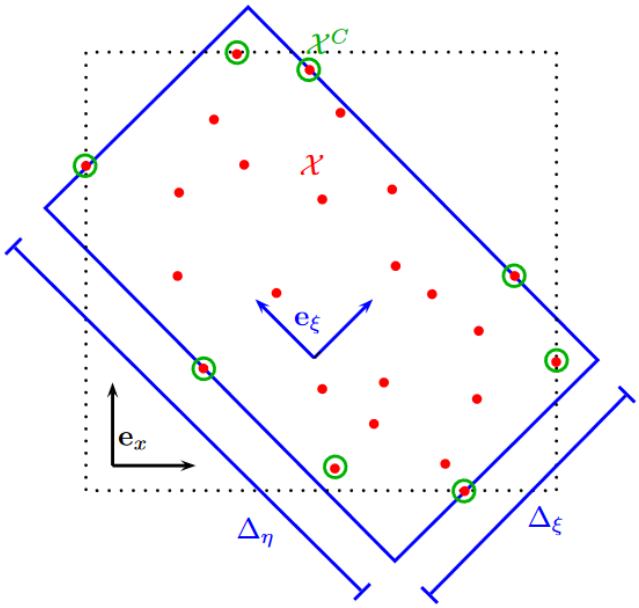


Figure 3.1.: Optimal oriented bounding box (solid blue lines) and axis aligned bounding box (AABB, dotted black lines) to a point set $\mathcal{X}$ in 2D, illustrating the used notation. Graphic from [CGM11].

- its rotation $R \in SO(d, \mathbb{R})$ and
- its extensions $\mathbf{\Delta} \in \mathbb{R}^d$,

where $SO(n, \mathbb{R}) = \{R \in \mathbb{R}^{d \times d} \mid R^T R = I_d = R R^T, det(R) = 1\}$ are all the orthogonal and real d-by-d-matrices. Given the definition, the optimal oriented bounding box is the solution to a constrained optimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{\Delta}, \mathbf{X} \in \mathbb{R}^3, R \in SO(3,\mathbb{R})} & \prod_{k=1}^{d} \mathbf{\Delta}_k \\ \text{s.t.} & -\frac{1}{2}\mathbf{\Delta} \leq RX_i - \mathbf{X} \leq \frac{1}{2}\mathbf{\Delta} \quad \forall i = 1, \ldots, N \end{array} . \tag{3.1}$$

The matrix R rotates the reference frame e_x onto e_ξ as shown in figure 3.1. In the rotated reference frame the bounding box is axis aligned.

An obvious condition for an optimal bounding box is, that each face contains at least one point of the point set $\mathcal{X}$. If not, the face can be moved towards the point set. Also, the convex hull $\mathcal{CH}(\mathcal{X})$ of $\mathcal{X}$ defines uniquely the optimal bounding box. The N_C vertices defining the convex hull are denoted as $\mathcal{X}_C$. The task of calculating the convex hull to a point set in three dimensions has the runtime complexity of $\mathcal{O}(N \log N)$ with N being the number of nodes in the point set [ORo98]. For the analysis of further conditions assume that the polygon is convex.

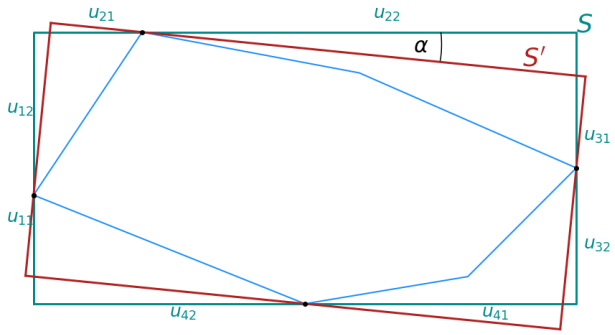


Figure 3.2.: Illustration of the rectangle S from proof of the lemma 1 and formation of rectangle S' by clockwise edge rotation of angle α.

Oriented Bounding Boxes in 2D

3D Oriented Bounding Boxes approximate the 3D simulation data, however, we start with an analysis of necessary conditions for 2D Oriented Bounding Boxes and their calculation based on the following lemma.

Lemma 1 (OBBs in 2D)**.** *A minimum-area rectangle circumscribing a convex polygon has at least one side flush with an edge of the polygon, that is to say a side contains every point of that edge. [FS75]*

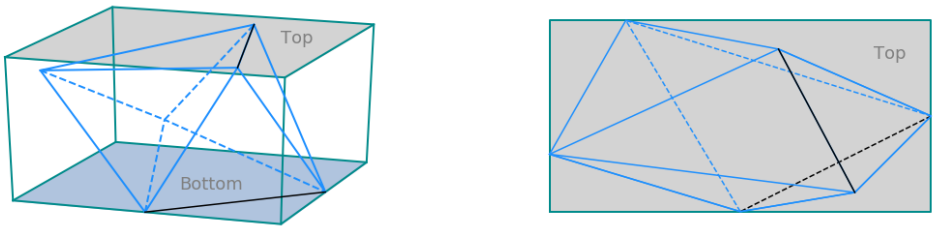
Proof. Assume, without loss of generality, that each side u_1, u_2, u_3, u_4 of the bounding box contains one point. Two adjacent sides can contain the same point.

Freeman and Shapira [FS75] have shown a proof by contradiction. Assume, that the bounding rectangle S does not contain an edge of the convex polygon and is of minimum area. Therefore, each side contains exactly one point and is divided into two segments $(u_i = u_{i1} + u_{i2}, i = 1, \ldots, 4)$ by the point they contain, as illustrated in figure 3.2. Define two rectangles S' and S'' with sides passing through the same points and still enclosing the convex polygon, such that the sides of S and S' form an angle α and the sides of S and S'' form an angle $-\alpha$. An $\alpha > 0$ exists, because every edge contains only one point and therefore can be turned.

Calculations for the area ΔA of S exceeding S' and the area $\Delta A'$ of S exceeding S'' lead to the results

$$\Delta A = K_1 + K_2 \sum_{i=1}^{4} (u_{i1}^2 - u_{i2}^2) \tag{3.2}$$

$$\Delta A' = K_1 - K_2 \sum_{i=1}^{4} (u_{i1}^2 - u_{i2}^2) \tag{3.3}$$



(a) Non-optimal bounding box of the polyhedron.

(b) Top view of the projection to the bottom face of the box.

Figure 3.3.: Illustration to case 2 from proof of lemma 2: Top and bottom face are flush with black edges of the polyhedron and the remaining faces are not flush with any edge. Dashed lines are non-visible edges of the polygon.

where K_1 and K_2 are both positive since $\alpha > 0$. The exact values for K_1 and K_2 can be found in [FS75].

At least one of the equations 3.2 and 3.3 must be positive. If $\Delta A > 0$, the rectangle S' is of smaller area than S; if $\Delta A' > 0$, the rectangle S'' is of smaller area than S. Therefore, the bounding rectangle S is not the minimum enclosing rectangle. □

Lemma 1 defines an algorithm for calculating optimal bounding boxes in 2D. Because one edge of the convex hull is flush with the enclosing rectangle, the optimal bounding box can be found by iterating over the edges. Hence, the runtime complexity is $\mathcal{O}(N_C)$, where N_C is the number of nodes defining the convex hull. This method is called *rotating calipers method* [Tou83].

Oriented Bounding Boxes in 3D

By applying lemma 1 we derive a necessary condition for oriented bounding boxes in three dimensions.

Lemma 2 (OBBs in 3D)**.** *A minimum-volume box circumscribing a convex polyhedron has at least two adjacent faces flush with edges of the polyhedron. [ORo85]*

Proof. Assume the contrary: There are fewer than two adjacent faces flush with edges of the polyhedron.

Let the faces of the box be called *Front*, *Back*, *Left*, *Right*, *Bottom* and *Top*. We show, that there must be four adjacent faces, all orthogonal to one, which form a ring and are not flush with any edges.

Case 1: If exactly one face F is flush with an edge, the faces orthogonal to F form a ring which is not flush with any edges.

Case 2: If exactly two nonadjacent faces are flush with an edge, as in figure 3.3, the two faces must be opposite to each other and the rest of the faces form a ring, which is not flush with any edges.

Case 3: More than two faces are flush with edges. In that case, two adjacent faces have to be flush with edges. Therefore, this case can be discarded.

Assume without loss of generality, that the faces *Front*, *Left*, *Back* and *Right* of the box, which form a ring, are not flush with any edges. The projection of the polyhedron to the *Bottom* plane of the box, is a convex polygon enclosed by a rectangle with none of the rectangle's sides flush with any edge of the projected polygon, as illustrated in figure 3.3b. As a consequence of lemma 1 the box is not minimal. The volume can be decreased by rotating the faces *Front*, *Left*, *Back* and *Right*, while keeping *Top* and *Bottom* fixed. □

3.2. Calculating 3D Oriented Bounding Boxes

The calculation of oriented bounding boxes in 3D is more challenging than for two dimensions. Although there is an algorithm that calculates an exact solution based on lemma 2, it has cubic runtime and we analyse different approximate solutions.

3.2.1. Exact Solution

Lemma 2 states that two adjacent faces of the convex hull $\mathcal{CH}(\mathcal{X})$ are flush with the oriented bounding box. Therefore, the exact algorithm iterates over all pairs of edges of the convex hull. While keeping the edges flush, it finds the rotation angle that minimizes the volume of the box. Latter minimization processes is described by [ORo85] and has linear runtime $\mathcal{O}(N_C)$ with N_C being the number of vertices defining the convex hull. The iteration over all pairs of edges leads to a runtime complexity in $\mathcal{O}(N_C^3)$ for the exact algorithm, whose minimal volume solution to the point set $\mathcal{X}$ is denoted as $BB_{opt}(\mathcal{X})$.

An algorithm with cubic runtime is not attractive for application. There are solutions that speed up the runtime, for example [Jyl15]. The author uses a graph search technique over the vertex graph of the convex hull, for which they showed an expected runtime complexity in $\mathcal{O}(N^{\frac{3}{2}}(\log N)^2)$ if the point sets has a uniform distribution of directions on their convex hull. It's worst case runtime lies in $\mathcal{O}(N^3 \log N)$.

3.2.2. PCA-Based Algorithm

The eigenvectors of the covariance matrix of the point set $\mathcal{X}$ are the principal components of the point set in 3D. The first eigenvector corresponding to the biggest eigenvalue indicates the most important direction in the data points and the number of eigenvectors, which are orthogonal to each other, equals the dimension of the point set.

Because the eigenvectors indicate the most important directions of the data, they can be used to approximate the optimal bounding box. The orthogonal eigenvectors of the covariance matrix define the orientations of the bounding box. Therefore, the resulting bounding box $BB_{PCA}(\mathcal{X})$ is the AABB in the coordinate system defined by the principal components of the point set $\mathcal{X}$. The methods based on the principal components are called PCA-based algorithms.

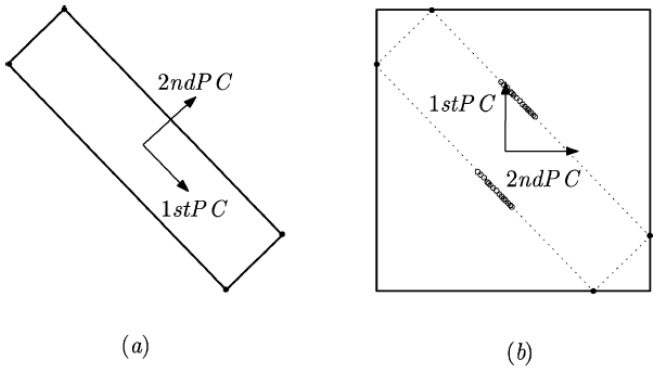


Figure 3.4.: Illustration of the dependence of the BB_{PCA} on the distribution of points. In (b) additional vertices are inserted turning the principal components by 45°.

The PCA-based algorithms are popular, because they have linear runtime in $\mathcal{O}(N)$, which is dominated by the calculation of the covariance matrix for the point set $\mathcal{X}$. The eigenvalue decomposition is fast, although having cubic runtime with respect to the dimension of the matrix, since the covariance matrix is of constant size 3×3. Two application of the PCA bounding boxes are for example OBBTree [GLM96] and BOXTREE [Bar+96]. The two hierarchical structures of bounding boxes are commonly used for data analysis, but the approximation of the optimal bounding boxes BB_{opt} with minimal volume is often rather poor.

To discuss the application of the PCA-based algorithm on our data, we want to have a closer look at the bounds of the PCA-based bounding boxes' volumes.

Bounds on the volume of PCA-based bounding boxes

The result of the PCA depends highly on the distribution of the points in $\mathcal{X}$, as illustrated in figure 3.4. In [Dim+09] they propose that the result can be improved if the convex hull's boundary is utilized to calculate the principal components, increasing the runtime to $\mathcal{O}(N \log N)$.

The authors of [Dim+09] studied lower and upper bounds on the general approximation factor κ_d of PCA-based bounding boxes to point sets in $\mathbb{R}^d$. The approximation factor for a fixed point set $\mathcal{X} \subset \mathbb{R}^d$ is the ratio of the volume of the PCA-based bounding box and the optimal solution $\kappa_d(\mathcal{X}) = \text{Vol}(BB_{PCA}(\mathcal{X}))/\text{Vol}(BB_{opt}(\mathcal{X}))$

$$\kappa_d = \sup\{\kappa_d(\mathcal{X}) | \mathcal{X} \subset \mathbb{R}^d, \text{Vol}(\mathcal{CH}(\mathcal{X})) > 0\}$$

The authors analyzed a series of approximation factors $\kappa_{d,i}$, where $0 \leq i \leq d$ is the dimension of the faces of the convex hull which are considered for PCA. This dimension depends on the representation of the convex hull.

As observed in figure 3.4 the approximation factor for $i = 0$ is unbounded, that is when utilizing points to define the convex hull. When considering a long thin rectangle

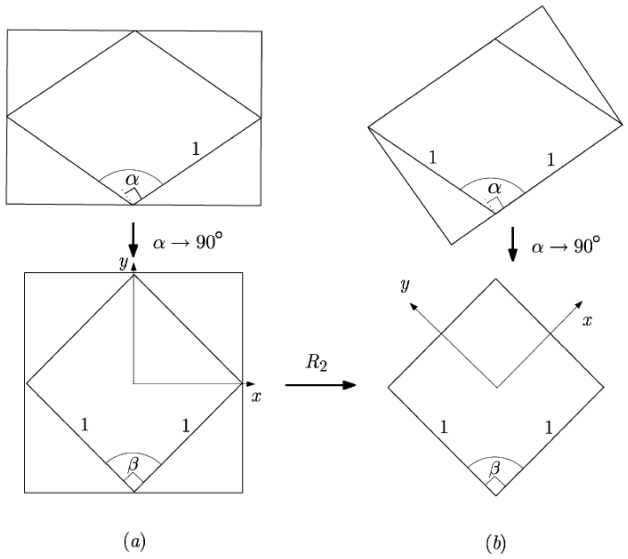


Figure 3.5.: Example showing the lower bound for the area of BB_{PCA} in $\mathbb{R}^2$. Illustration of the bounding boxes (a) BB_{PCA} and (b) BB_{opt}, from [Dim+09].

and inserting vertices in the middle of the long edges, the principal components are turned by 45°. The same construction can be done in higher dimensions as well, leading to the following lemma.

Lemma 3. $\kappa_{d,0} = \infty$ *for any* $d \geq 2$.

Instead of considering points ($i = 0$) the continuous version of PCA can be applied to the dense set of all points in the convex hull $\mathcal{CH}(\mathcal{X})$ ($i = d$) or on its boundary ($i = d-1$). The continuous PCA calculates the coefficients of the covariance matrix using integrals instead of finite sums. [Dim+09] gives a nice summary of the continuous PCA applied to triangulated surfaces, a detailed version can be found in [GLM96] and [VSR01].

The authors of [Dim+09] have shown two lower bounds for the approximation factors in $\mathbb{R}^2$ and $\mathbb{R}^3$. The sketch of the illustrative proofs can be found in the appendix A.1:

- $\kappa_{2,1} \geq 2$ and $\kappa_{2,2} \geq 2$
- $\kappa_{3,2} \geq 4$ and $\kappa_{3,3} \geq 4$

The proofs show, that PCA-based bounding boxes tend to fail, if the eigenvalues are close to each other and the side lengths of the optimal bounding boxes are almost equal to each other. In that case there is no strong principal direction, since the point set $\mathcal{X}$ is distributed similarly in all directions. Figure 3.5 illustrates the problem in two dimensions.

The authors of [Dim+09] have shown three upper bounds for the approximation factors in $\mathbb{R}^2$ and $\mathbb{R}^3$, whose proofs can be found in their publication:

- $\kappa_{2,1} \leq 2,737$ when computing the BB_{PCA} over the boundary of the convex hull $\mathcal{CH}(\mathcal{X})$
- $\kappa_{2,2} \leq 2,104$ when computing the BB_{PCA} over the convex hull $\mathcal{CH}(\mathcal{X})$
- $\kappa_{3,3} < 7.81$ when computing the BB_{PCA} over the convex hull $\mathcal{CH}(\mathcal{X})$

3.2.3. HYBBRID-Algorithm

The optimal bounding box is defined uniquely by its center $\mathbf{X} \in \mathbb{R}^d$, its rotation $R \in SO(d, \mathbb{R})$ and its extensions $\mathbf{\Delta} \in \mathbb{R}^d$. If $d = 3$ let $\mathbf{\Delta} = (\Delta_\xi, \Delta_\eta, \Delta_\zeta)$.

The direct definition of the optimal bounding box as the optimization problem (3.1) leads to a constrained optimization problem. A redefinition of (3.1) as in [CGM11] allows its analysis as an unconstrained optimization over $SO(3, \mathbb{R})$

$$\min_{R \in SO(3,\mathbb{R})} f(R),$$

where $f(R)$ is the volume of the AABB of the point set $\mathcal{X} = \{X_i \mid i = 1, \ldots, N\}$ rotated by R. Its evaluation, which includes multiplication by the 3×3-matrix R and calculating the minimum and maximum three times, has linear runtime. When completely formulating f this leads to

$$\min_{R \in SO(3,\mathbb{R})} \begin{pmatrix} \min\limits_{\Delta \in \mathbb{R}^3, \mathbf{X} \in \mathbb{R}^3} & \Delta_\xi \Delta_\eta \Delta_\zeta \\ \text{s.t.} & -\frac{1}{2}\mathbf{\Delta} \leq RX_i - \mathbf{X} \leq \frac{1}{2}\mathbf{\Delta} \quad \forall i = 1, \ldots, N \end{pmatrix},$$

where the operator $\leq$ is applied componentwise [CGM11].

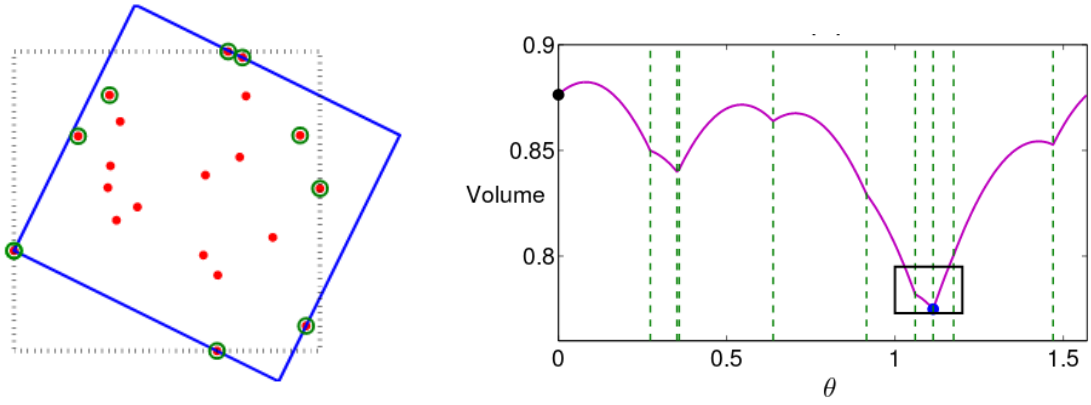
When considering this approach the points in $\mathcal{X}$ are considered to be the vertices of a convex polyhedron, since it makes the evaluation of f faster giving the same results.

The function $f : SO(3, \mathbb{R}) \mapsto \mathbb{R}$ is not differentiable. Especially in all the rotations, where the AABB has two adjacent faces flush with edges of the polyhedron, the function f is not differentiable. Nevertheless, many of those rotations yield local minima, and consequently also the rotation defining the optimal bounding box. Figure 3.6 illustrates the function f for a two-dimensional example.

Therefore, the authors of [CGM11] considered a derivative-free optimization method to avoid instabilities. Since there are many local minima, the optimization method is a combination of global search (Exploration of the search space $SO(3, \mathbb{R})$) and in promising regions local search (Exploitation). The algorithm's name Hybrid Bounding Box Rotation Identification (HYBBRID) derives from the combination of Exploration and Exploitation steps.

Exploitation

The Nelder-Mead simplex algorithm [NM65] optimizes using direct search. The algorithm considers simplices, whose d+1 vertices ($R_1, \ldots, R_{d+1} \in SO(3, \mathbb{R})$) are the currently considered samples in the search space $SO(3, \mathbb{R})$.



(a) (b)

Figure 3.6.: Two-dimensional example of the function $f : SO(3, \mathbb{R}) \mapsto \mathbb{R}$, the volume of the minimal bounding rectangle rotated by an angle θ from 0° to 90° to the point set demonstrated in (a). The dashed lines in (b) correspond to angles, where the bounding rectangle is flush with a side of the convex hull. Figure from [CGM11].

Given an initial simplex of affinely independent points, in each iteration the worst vertice is replaces by its reflection R_{new} through the centroid of the d remaining points, if R_{new} gives sufficiently good results (reflection of the simplex). If this newly created rotation R_{new} is better than the current best rotation, a bigger step in the same search direction is tested (expansion of the simplex). In case the new rotation R_{new} is not satisfactory, the simplex is shrunk, since the optimal solution may lie inside the simplex (contraction or reduction). [CGM11] gives a detailed explanation of how the simplices change in each iteration.

The result of the Nelder-Mead simplex algorithm depends highly on the initial simplex and the algorithm is not guaranteed to converge to a stationary point [McK98]. Therefore, the idea is to take the best result from several runs using different initializations and by that explore the search space.

Exploration

After every optimization of the simplices using the Nelder-Mead algorithm (Exploitation) a new generation of simplices $\mathcal{A}$ is created (Exploration). During the exploration step we want to explore new regions in the search space, but at the same time introduce some correlation between the samples to concentrate on promising candidates [CGM11]. Genetic algorithms [Gol89; Hol75] achieve that in the style of evolutionary biology inspired by Darwin's theory of evolution. Each individual of the population $\mathcal{A}$ is assigned a fitness, and individuals with a higher fitness contribute most to the next generation. This principle is known as survival of the fittest. In summary, the operators of a genetic

algorithm are selection, crossover of the fittest samples and mutation to create fitter generations. By that we expect to optimize the initial simplex for the Nelder-Mead algorithm and therefore improve results.

For the HYBBRID algorithm a population $\mathcal{A}$ with M initial simplices, which contain four rotation matrices each, is considered. The M initial simplices A_i, $i = 1, \ldots, M$ are chosen randomly in the beginning, leading to a total of $4 \cdot M$ rotational matrices $R_{i,j}$ for $i = 1, \ldots, M, j = 1, \ldots, 4$. The orthogonal rotation matrices are obtained by applying QR-factorization to random 3×3 matrices [CGM11].

The authors of [CGM11] defined the HYBBRID algorithm based on the genetic exploration of the search space and exploitation via the Nelder-Mead simplex algorithm in six steps:

1. **Initialization**: Select M random initial simplices, each made up of 4 rotation matrices in $SO(3, \mathbb{R})$

2. **Selection**: Calculate the fitness $min_{i=1,\ldots,4}\ f(R_{i,j})$ for $j = 1, \ldots, M$ of the simplices and discard the $\frac{M}{2}$ worst ones. The remaining $\frac{M}{2}$ simplices are randomly sampled into 4 groups $\mathcal{A}_1^I, \mathcal{A}_2^I, \mathcal{A}_1^{II}, \mathcal{A}_2^{II}$, of the size $\frac{M}{2}$, allowing resampling of simplices.

3. **Crossover I**: Apply mixing crossover to $\mathcal{A}_1^I$ and $\mathcal{A}_2^I$ by selecting a pair of simplices $A \in \mathcal{A}_1^I$ and $A' \in \mathcal{A}_2^I$ (the parents) to generate one new simplex (the child). Each vertex of the child is selected from one of the corresponding parent-simplices, selecting the better parent with a higher probability. Since their are $\frac{M}{2}$ different pairs, $\frac{M}{2}$ new simplices are created.

4. **Crossover II**: Apply an affine combination crossover to $\mathcal{A}_1^{II}$ and $\mathcal{A}_2^{II}$ by selecting the parents A and A' as above to generate one new simplex (the child). The four vertices of the child A^c are chosen via an affine combination $A_j^c = \lambda A_j + (1 - \lambda) A'_j$. Lambda is selected such that the fitter parent-simplex has a higher weight.

5. **Mutation**: Apply K iterations of the Nelder-Mead algorithm to the M newly created simplices in the crossover processes generating a new generation of $\mathcal{A}$.

6. **Stopping Criterion**: Repeat the Selection - Crossover - Mutation process until the fitness of the best simplex in $\mathcal{A}$ does not change significantly any more, but a at least it_{min} and at most it_{max} times.

The authors of [CGM11] point out the good approximation quality of $BB_{HYBBRID}$ of the optimal bounding box BB_{opt} obtained with O'Rourke's exact algorithm, if the population size M and the number of iterations of Nelder-Mead K are high enough. They stopped the algorithm if the best rotational matrix doesn't improve its fitness at least 1% during the last 5 iterations. Since the HYBBRID algorithm includes metaheuristic procedures, no runtime estimate can be given. The experimental results in [CGM11] show a significantly lower runtime than O'Rourke's algorithm.

3.2.4. Adapted HYBBRID Algorithm

The result of the exploitation step, during which the Nelder-Mead simplex algorithm is applied, depends highly on the quality of the initial simplices. A good initial guess can therefore improve the stability and the runtime of the HYBBRID algorithm significantly.

Since the time scale is sampled densely for car crash simulations, the rotations of the oriented bounding boxes from subsequent timesteps are similar. Hence, one vertex of the first simplex is set to be the best rotation matrix from the previous timestep. The change allows reduction of the population size M by half from 30 to 15 and of the stopping criterion to less than 1% improvement in the last 4 instead of 5 iterations.

Utilizing the implementation from the authors the best rotation matrix could be discarded during the crossovers. The implementation was changed such that the fittest individual always stays in the next generation to account for the smaller population size M.

Since the optimal oriented bounding box has two faces flush with edges of the convex polyhedron as shown in lemma 2, the oriented bounding box could move abruptly from timestep to timestep, although the solutions are optimal. This can occur if the reference edges change and the point cloud is not densely sampled.

To correct possible instabilities in timesteps that immediately succeed each other the adapted version of the algorithm stabilizes the bounding boxes if necessary. If for three consecutive timesteps $t-1, t$ and $t+1$ the mean squared error of the bounding boxes BB_{t-1} and BB_{t+1} is less than 2% of the mean squared error of BB_t and its neighboring boxes, the rotation in t is replaced.

To stabilize the bounding box in timestep t, a new rotation $R_{t,new}$ is defined by the average of the rotations' Euler angles from $t-1$ and $t+1$. The bounding box is replaced by the axis aligned bounding box in the coordinate system defined by $R_{t,new}$.

3.3. Application of the Algorithms

Given the simulation data as described in chapter 2 an oriented bounding box is calculated for every car part $p \in P$ at every timestep t and for all the simulations $sim \in SIM$. The oriented bounding boxes for different timesteps are considered as a time series for each car part. Therefore, it is necessary to ensure the boxes' vertex-to-vertex correspondence. This does not effect the information value, but the order of the vertices, which is crucial for the functionality of the neural network. For the initial timestep, which is equal for all simulations, a fixed order has been defined, and the boxes' vertices in the following timesteps have been considered in the same order.

The bounding box should summarize the general deformation of the part and allow distinction between different deformation modes. Especially, the boxes allow comparison of a component's deformation that is discretized at different resolutions or has geometry changes and therefore cannot be compared point-wise. To achieve that goal the approximations need to be of high quality and, in particular, stable, since we want to approximate time series. At the same time, the calculations should be realizable in a

Algorithm	Relative Volume
PCA, point based	2.525
PCA, convex hull based	3.417
HYBBRID	1.00007
HYBBRID + Adaptions	1.00006
Exact	1

(a) Comparison of the relative volume to the minimal volume of the different algorithms for front beams.

Algorithm	Program	Runtime for one simulation
PCA-based	Python	3 minutes
Exact	Matlab	ca. 100 h*
HYBBRID	Matlab	2.6 h
HYBBRID + Adaptions	Matlab	53 minutes

(b) Runtimes of the different algorithms for oriented bounding box calculation for all car parts and timesteps of a simulation without parallelization.
*estimate based on calculation of BB_{opt} for four parts.

Table 3.1.: Quality and runtime of algorithms for oriented bounding box calculation.

reasonable time to make the analysis applicable during the simulation runs.

Quality

The relative volumes compared to the minimum volume of the oriented bounding boxes for the different algorithms are listed in table 3.1a. Since the calculation of the exact bounding boxes is time-consuming only the front beams have been considered for the comparative evaluation. Additionally, figure A.3 in the appendix illustrates the different bounding boxes for the left front beam at different timesteps.

Table 3.1a displays that the results of the HYBBRID algorithm and the adapted HYBBRID algorithm are close to the exact solution and stable over time.

As mentioned earlier, the exact oriented bounding boxes $BB_{opt}(\mathcal{X})$ sporadically jump since two edges of the convex polygon are flush with the bounding boxes. For the TRUCK dataset unstable bounding boxes can be observed for round car parts, which include parts from the wheels. Those parts making up the wheels have been excluded from the analysis. In a few examples the instabilities are noticeable for other car parts. The enhanced initialization with the rotation matrix from the previous timestep reduces the number of instabilities. All remaining cases are successfully eliminated by the adapted HYBBRID algorithm's stabilization process.

The volumes of the PCA-based bounding boxes are 3 times bigger than the minimum volume. Since the car parts are densely sampled, the triangulation of the convex hull's boundary could not improve the bounding box volumes. The bounding boxes BB_{PCA} cannot sufficiently represent the car part's deformation, because of the bad approximation and their instability as commented in section 3.2.2.

Runtime

The runtime of the exact algorithm in Matlab (implementation by [CGM11]) is approximately 100 h for one sample, that means calculating BB_{opt} for all the parts and timesteps for one car model, see table 3.1b. This estimate is based on the runtime for calculating the exact bounding boxes for the front beams. Although the calculation of the OBBs can easily be parallelized, because the calculations of the boxes do not depend on each other, the application of the exact algorithm is not feasible.

Calculating the PCA-based bounding boxes is fast. However, since the volume of BB_{PCA} is 2 to 3 times bigger than the optimal volume as well as unstable, the deformations are not sufficiently well represented and the algorithm is discarded.

The HYBBRID algorithm is 35 times faster than the exact algorithm for the TRUCK dataset while producing results of similar quality. However it is still time consuming. The adapted HYBBRID algorithm reduces the runtime by an additional factor of three, because the population size has been decreased and the algorithm converges faster due to the good initialization.

Therefore, the adapted HYBBRID algorithm is chosen to preprocess the simulation data into bounding boxes. The final model does not require oriented bounding boxes for all parts and timesteps, since the network works with a subset of parts and timesteps. Additionally, the calculation of bounding boxes can be parallelized, since the HYBBRID algorithm separately considers boxes from different parts and simulations. Hence, for the final usecase the runtime can be considerably reduced.

3.4. Information Retrieval from OBBs

Figure A.4 in the appendix illustrates the preprocessing result, showing the bounding boxes of selected parts of the TRUCK dataset. Additionally, for the left front beam (part-id 2000001) of the TRUCK data set, which initializes the bifurcation BI1, we notice that the HYBBRID-bounding boxes distinguish between the two branches of the bifurcation, see figure A.2 in the appendix.

The afore mentioned figures show the original bounding boxes, including translation, rotation and scale. The oriented bounding boxes further allow consideration of translation, rotation and scale of the car components individually. Normally the rigid transform, composed of the translation and rotation, is estimated using a few fixed points, so called follow-up points, for each component [SW94]. The quality of the rigid transform estimation depends highly on the hand selected follow-up points. We are interested in removing the rigid transform of the car parts before the analysis because it is many times induced by other parts' plastic deformation.

When utilizing oriented bounding boxes the use of follow-up points is not necessary any more. The parametrization of the deformation into translation, rotation and scale allows not only analysis of the original bounding boxes, but also exclusively analyzing the scale. For this purpose the bounding boxes are shifted into the origin and rotated to the standard coordinate axes. The rotation matrix is available as a result of the adapted

HYBBRID algorithm. This allows detecting interesting characteristics in the car parts' deformations that are not translation or rotation based.

4. Model: LSTM Autoencoder

The analysis of simulation runs is tedious because the results are high dimensional and the majority of the car parts' deformations are similar. However, the car parts, whose deformations show different characteristics are of special interest. Those car parts are analyzed in detail to select parameters producing superior, safer and more stable deformation results.

The selected architecture should therefore assist in finding the deformation characteristics that distinguish the simulation runs. Additionally, we want the analysis to be available in-situ and as soon as possible, to run simulations with different parameters quickly.

[AIG19] applied the LSTM autoencoder [SMS15] successfully on the car crash simulation dataset TRUCK, which was preprocessed by spectral descriptors based on Laplace Beltrami Operators. We take up the general architecture of the paper and apply it to the TRUCK data represented as oriented bounding boxes, which have a lower dimension than the most significant spectral coefficients of the Laplace Beltrami Operators and allow decomposition of the deformation into rotation, translation and scale. We additionally explore the postprocessing of the hidden representations, which allows straight forward comparison of different models.

The main components of the LSTM autoencoder, its extension and postprocessing are presented in this chapter.

4.1. Autoencoder

Neural Networks are useful tools for supervised as well as unsupervised learning because of their high approximation power. Autoencoders are a type of unsupervised learning algorithm which have the principle purpose of learning low dimensional characteristic features of the input that allow its recreation it as well as possible. Therefore, it is an alternative to dimension reductions like the linear or kernel PCA, Isomap or Diffusion Maps [GBC16].

An autoencoder is made up of two parts: An encoder neural network whose output is the low dimensional feature representation $\mathbf{h} \in \mathbb{R}^{d_h}$ of the d_n-dimensional input and a decoder neural network which, given the low dimensional features, recreates the input vector. The encoder and decoder are trained at the same time to approximate the identity function. In case of undercomplete autoencoder, it holds $d_h < d_n$ and we expect that the low dimensional representation captures the most prominent features of the training data and allows distinction between the samples [GBC16].

In practice, the user is generally not interested in the output of the decoder, the

recreated input, but in the low dimensional feature representations, which sometimes are also called hidden representations and show the most relevant aspects that distinguish the input from the rest of the data.

The autoencoder is generally trained by stochastic gradient descent, for which there are no error bounds. Consequently, as all types of neural networks do, an autoencoder requires a high amount of training data and its validation with unseen data is inevitable.

An advantage of autoencoders over standard dimension reduction methods is their generalization performance, because they learn an encoder. The encoder is a function, that is quickly evaluated and describes the manifold. Therefore, new data points' embeddings don't have to be obtained by interpolation.

Because the different car crash simulations have a lot of information in common and differences can only be detected in particular car parts, autoencoders have been chosen to distinguish between those deformation modes and locate the car parts they appear in and affect.

4.2. Long Short-Term Memory

The simulation dataset TRUCK consists of time sequences $S = \{s_1, \ldots, s_{T_{total}}\}$ with $s_t \in \mathbb{R}^n$ for $t = 1, \ldots, T_{total}$ describing a rectangular box which translates, rotates and scales over time. The information from one to the next timestep is highly correlated and a neural network should take advantage of this characteristic, because availability of correlated information means that some analysis blocks could be reused for all timesteps instead of training them various times for similar purposes.

This method, called parameter sharing over time, has been implemented in the standard recurrent neural networks (RNN) [RHW86] and reduce the amount of trainable parameters. Another famous and successful application of parameter sharing is the spacial convolution parameter, that is applied at different locations to the input [LeC+89].

Parameter sharing over time means that the architecture applies for every timestep the same weights to analyze and sum up the input vector into a hidden representation h'_t. A standard RNN incorporates the information from the previous timestep $t-1$ via a weighted sum $h_t = h'_t + Wh_{t-1}$. Generally, the information from timestep $t-2$ has been included in the summary vector $h_{t-1} = h'_{t-1} + Wh_{t-2}$. Therefore, information from all previous timesteps is considered in a recursive manner. Nevertheless, the information from timestep $t-l$ is weighted by W^l, which, because of the multiplication of the matrices, leads to vanishing or exploding gradients during gradient descent, depending on whether the eigenvalues of the weight matrix W are larger or smaller than 1 [Hoc91]. That aggravates learning long term dependecies [Hoc+01]. [GBC16] gives a detailed explanation of the vanishing/exploding gradient problem and presents some architectural solutions to learning long-term dependencies over time while keeping the general structure of the RNN.

Long short-term memory (LSTM) [HS97; Gra13] is a method for handling sequences inside an RNN and applies parameter sharing over time. LSTM circumvents the vanishing/exploding gradient problem by incorporating information from previous timesteps

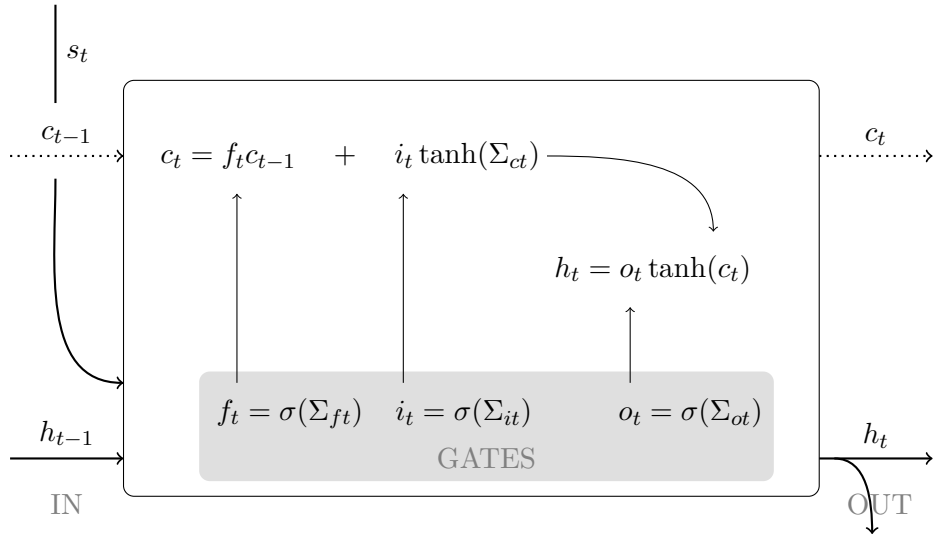


Figure 4.1.: LSTM cell, where $\Sigma_{\bullet t} = W_{s\bullet} s_t + W_{h\bullet} h_{t-1} + b_\bullet$. $\bullet$ indicates the corresponding gate (f, i, o) or the cell state (c). Note that the weights of the sums are shared over time.

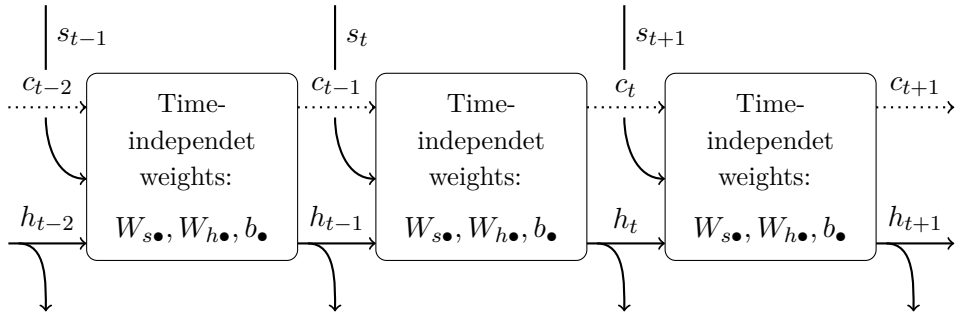


Figure 4.2.: Unfolded LSTM layer showing the interaction of the neighboring LSTM cells. The hidden vector of a cell is fed to the subsequent LSTM cell in the same layer and is forwarded to the next layer.

over sums and not multiplications.

For every analyzed timestep an LSTM cell or unit calculates a cell state c_t as an internal memory, which is updated for each timestep t. Let the input dimension to the LSTM unit be m. The input $s_t \in \mathbb{R}^m$ and the hidden state from the previous timestep h_{t-1} are summed up by sigmoidal input, output and forget gates, which take values between 0 and 1 and thereby control the cell's activation. The cell, whose internal and output variables are k-dimensional, outputs a hidden state $h_t \in \mathbb{R}^k$.

Having h_0 and c_0 initialized with zeros, for the following timesteps the calculations inside the cells, as implemented in Keras [Cho+15] and defined in [HS97], are

$$
\begin{aligned}
i_t &= \sigma(W_{si} s_t + W_{hi} h_{t-1} + b_i), \\
f_t &= \sigma(W_{sf} s_t + W_{hf} h_{t-1} + b_f), \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{sc} s_t + W_{hc} h_{t-1} + b_c), \\
o_t &= \sigma(W_{so} s_t + W_{ho} h_{t-1} + b_o), \\
h_t &= o_t \tanh(c_t).
\end{aligned}
$$

The formulas are illustrated graphically in figure 4.1. An LSTM layer, that is unfolded over time, can be imagined as consisting of as many connected LSTM cells as input timesteps, see figure 4.2. Since all the LSTM cells have the same weights the implementations use only one cell.

If the input dimensions of the sequence is m, the number of features per timestep, and the hidden dimensions k, which is the variable that can be chosen for an LSTM layer, the sizes of the vectors and matrices are:

- $s_t \in \mathbb{R}^m$ input vector to the LSTM unit
- $h_t \in \mathbb{R}^k$ hidden state vector also known as output vector of the LSTM unit
- $i_t, f_t, o_t \in \mathbb{R}^k$ input, forget, output gate's activation vector
- $c_t \in \mathbb{R}^k$ cell state vector
- $W_{s\bullet} \in \mathbb{R}^{k \times m}$: trainable weight matrices applied to input sequence
- $W_{h\bullet} \in \mathbb{R}^{k \times k}$: trainable weight matrices applied to hidden states from previous timestep
- $b_\bullet \in \mathbb{R}^k$: trainable bias vectors

In sum, the number of trainable parameters for one LSTM layer depends only on the input and output dimensions, m and k. For the selected implementation [Cho+15]

$$
4\underbrace{km}_{W_{s\bullet}} + 4\underbrace{kk}_{W_{h\bullet}} + 4\underbrace{k}_{b_\bullet} = 4k(1 + m + k)
$$

matrix and bias weights are trainable. This number does not depend on the length of the input time sequence, since the parameters are shared over time.

For multilayer LSTM networks, various LSTM layers are stacked and the hidden state vectors from all timesteps can be fed to another LSTM layer whose input dimension is the hidden dimension from the previous layer, as clarified in figure 4.2.

4.3. LSTM Autoencoder

For an LSTM autoencoder the encoder and decoder of the standard autoencoder are substituted by LSTM layers to specialize it on the analysis of time sequences [SMS15]. Assume for the definition of the architecture that we analyze only one car part $p \in P$. That means each sample $s_t \in \mathbb{R}^n$ for $t = 1, \ldots, T_{total}$ of the time series $S = \{s_1, \ldots, s_{T_{total}}\}$ is the rectangular bounding box of the car part p and $n = 24$. The input contains the first T_{in} timesteps ($S_{in} = \{s_1, \ldots, s_{T_{in}}\}$) of the simulation, so that the analysis provides results during the simulation runs.

The encoder LSTM sums up the time sequence S_{in} into a low-dimensional representation, taking advantage of the parameter sharing over time and detecting the features that distinguish the timesteps, since the general structure is the same in all timesteps. The dimensionality of the encoder LSTM units is the desired size l of the low-dimensional representation. The learned hidden representation of the LSTM autoencoder is then defined as the hidden vector of the last LSTM unit of the encoder layer $\mathbf{h} = h_{T_{in}} \in \mathbb{R}^l$.

This hidden vector is input to all the LSTM cells of the decoder layer, whose hidden vectors h'_t might not have the desired dimension n of the input. They are fed through a fully connected layer $out()$, outputting the n-dimensional $o_t = out(h'_t) \in \mathbb{R}^n$, for $t = 1, \ldots, T_{in}$. The output $S_{reconstruct} = \{o_1, o_2, \ldots, o_{T_{in}}\}$ tries to regenerate the input sequence S_{in}.

Figure 4.3 illustrates the standard LSTM autoencoder, which has been explained above.

During training, gradient descent optimizes all trainable weights while minimizing the mean squared error (MSE) over the input sequence S_{in} and the reconstructed sequence $S_{reconstruct}$

$$L = \frac{1}{T_{in}} \sum_{t=1}^{T_{in}} MSE(o_t, s_t).$$

The mean squared error of the true value s_t and the predicted value o_t, both n-dimensional, is

$$MSE(o_t, s_t) = \frac{1}{n} \sum_{i=1}^{n} ((o_t)_i - (s_t)_i)^2.$$

For analysis we are generally interested in the l-dimensional representation $\mathbf{h} \in \mathbb{R}^l$ of the input time sequence. Hence, l should be smaller than the size of the input sequence $T_{in} \cdot n$ and, if the training was successful, $\mathbf{h}$ represents the important features of the input time sequence $S_{T_{in}}$, such that its reconstruction is possible.

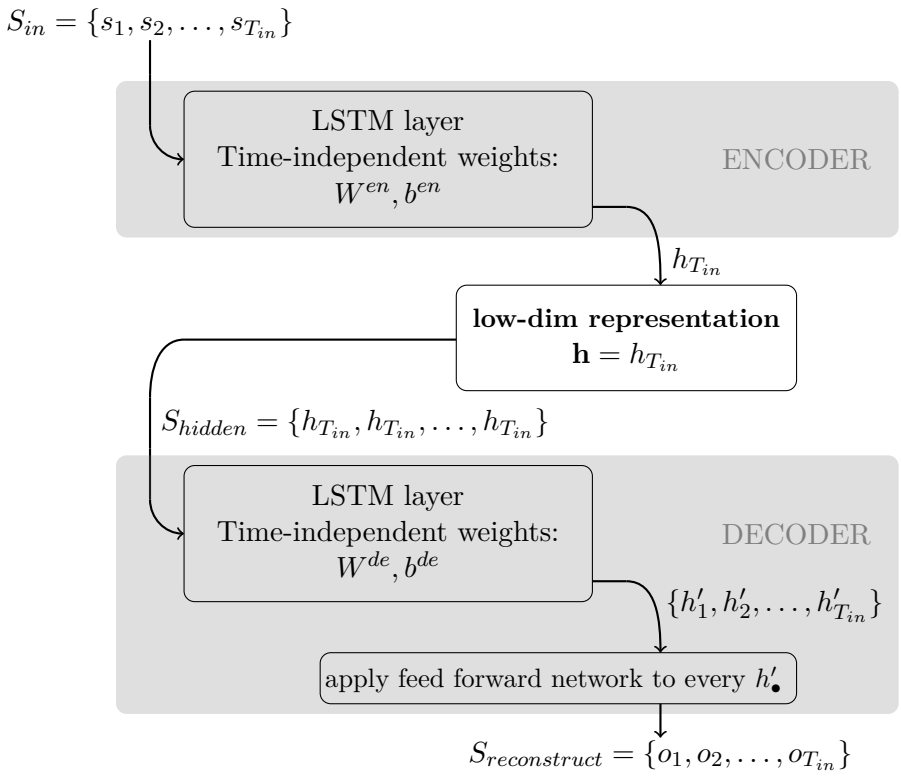


Figure 4.3.: LSTM autoencoder. The last hidden vector $h_{T_{in}}$ of the encoder layer is utilized as the low dimensional representation $\mathbf{h}$, which is input to all the LSTM units of the decoder. By applying a time independent feed-forward network to the m-dimensional hidden vectors $\{h'_1, h'_2, \ldots, h'_{T_{in}}\}$ of the decoder, the output sequence $S_{reconstruct}$ is created.

4.3.1. Composite Model

The LSTM autoencoder with one decoder reconstructs the input time series S_{in}. Hence, it learns features distinguishing between deformations taking place in the first T_{in} timesteps. The deformation modes in the future timesteps, especially for deformations that commence at T_{in}, are similarly if not more interesting and should also be represented in the low dimensional features.

Therefore, we try an autoencoder with two different decoders [SMS15]. The first decoder reconstructs the input sequence S_{in} like the standard LSTM autoencoder and a second decoder predicts the oriented bounding box positions in the future timesteps $S_{pred} = \{s_{T_{in}+1}, \dots, s_{T_{total}}\}$.

The so called composite model is also trained with gradient descent minimizing the sum of the mean squared errors over the reconstruction of S_{in} and prediction of S_{pred}

$$L' = \frac{1}{T_{in}}\Big(\sum_{t=1}^{T_{in}} MSE(o_t, s_t)\Big) + \frac{1}{T_{total} - T_{in}}\Big(\sum_{t'=T_{in}+1}^{T_{total}} MSE(o_{t'}, s_{t'})\Big).$$

It is possible to weight the losses of the prediction and reconstruction decoder differently

$$L'_w = \frac{w_1}{T_{in}}\Big(\sum_{t=1}^{T_{in}} MSE(o_t, s_t)\Big) + \frac{w_2}{T_{total} - T_{in}}\Big(\sum_{t'=T_{in}+1}^{T_{total}} MSE(o_{t'}, s_{t'})\Big),$$

where $w_1, w_2 > 0$.

In contrast to the reconstructed input sequence, the prediction should not be discarded. It can give a first estimate of the simulation result, although it does not substitute the actual simulation result. In addition to the analysis of the hidden representation, the prediction helps discard or abort ongoing simulations and pick new simulation parameters for new runs.

Furthermore, the low dimensional embedding is expected to be more significant with respect to the detection of bifurcations and different deformation modes in the future, since the low dimensional features are also trained to predict future deformation.

4.4. Postprocessing and Visualization of Results

The set of the hidden representations $\mathcal{H} = \{\mathbf{h}^{(p,sim)} \mid \forall sim \in SIM'\} \subset \mathbb{R}^l$ for a fixed car part $p \in P$ and simulations $SIM' \subseteq \{sim_1, sim_2, \dots\}$ contains useful information for, on one hand, an analysis of the car part p and on the other hand, controlling whether the autoencoder learns significant features.

Generally, the hidden dimension is higher than 3. Therefore, we utilize the t-SNE dimension reduction algorithm (t-distributed stochastic neighbor embedding) [MH08] to embed the hidden representation in two dimensions for visualization purposes. t-SNE visualizes high dimensional data by finding a stochastic, non-linear 2D-embedding $\mathcal{H}_{emb} \subset \mathbb{R}^2$ of a high-dimensional point set $\mathcal{H} \subset \mathbb{R}^l$.

The algorithm consists of 2 steps:

1. t-SNE defines a probability distribution Q based on Gaussian kernels over the relationships between two points in $\mathcal{H}$, such that similar points have a high probability, whereas dissimilar points have a low probability. The size of the neighborhood or the bandwidth of the Gaussian kernel is adapted to the local density of the point set, therefore the sizes of the neighborhoods are variable.

2. Given the probability distribution Q over the original d-dimensional data, a second probability distribution Q' over the 2-dimensional points is defined. Here a heavy-tailed Student t-distribution is chosen, defining the similarity between points. This prevents the crowding problem, where points tend to get crowded in low-dimensional space due to the curse of dimensionality. The Kullback–Leibler divergence $KL(Q \,||\, Q')$ of the distribution Q' from Q is minimized using gradient descent with respect to the locations of the points in 2D.

Because of the variable size of the neighborhoods, t-SNE identifies clusters even when they are not linearly separable. Likewise, the method yields meaningful results for visualizing the hidden representations of more than one part in two dimensions. On the other hand, also because of the irregular neighborhood sizes, the spreads and distances of clusters can be misleading.

If the autoencoder learns a good l-dimensional representation, we expect to be able to detect the already known bifurcation BI1 and possibly new patterns in the 2D-embeddings of low-dimensional hidden representations $\mathbf{h}^{(p,sim)}$. Therefore, the 2D embedding helps determine whether the autoencoder learned a meaningful embedding and helps detect different deformation modes of the car parts.

Figure 4.4 summarizes the whole pipeline of preprocessiong, modelling and postprocessing, illustrating the notation.

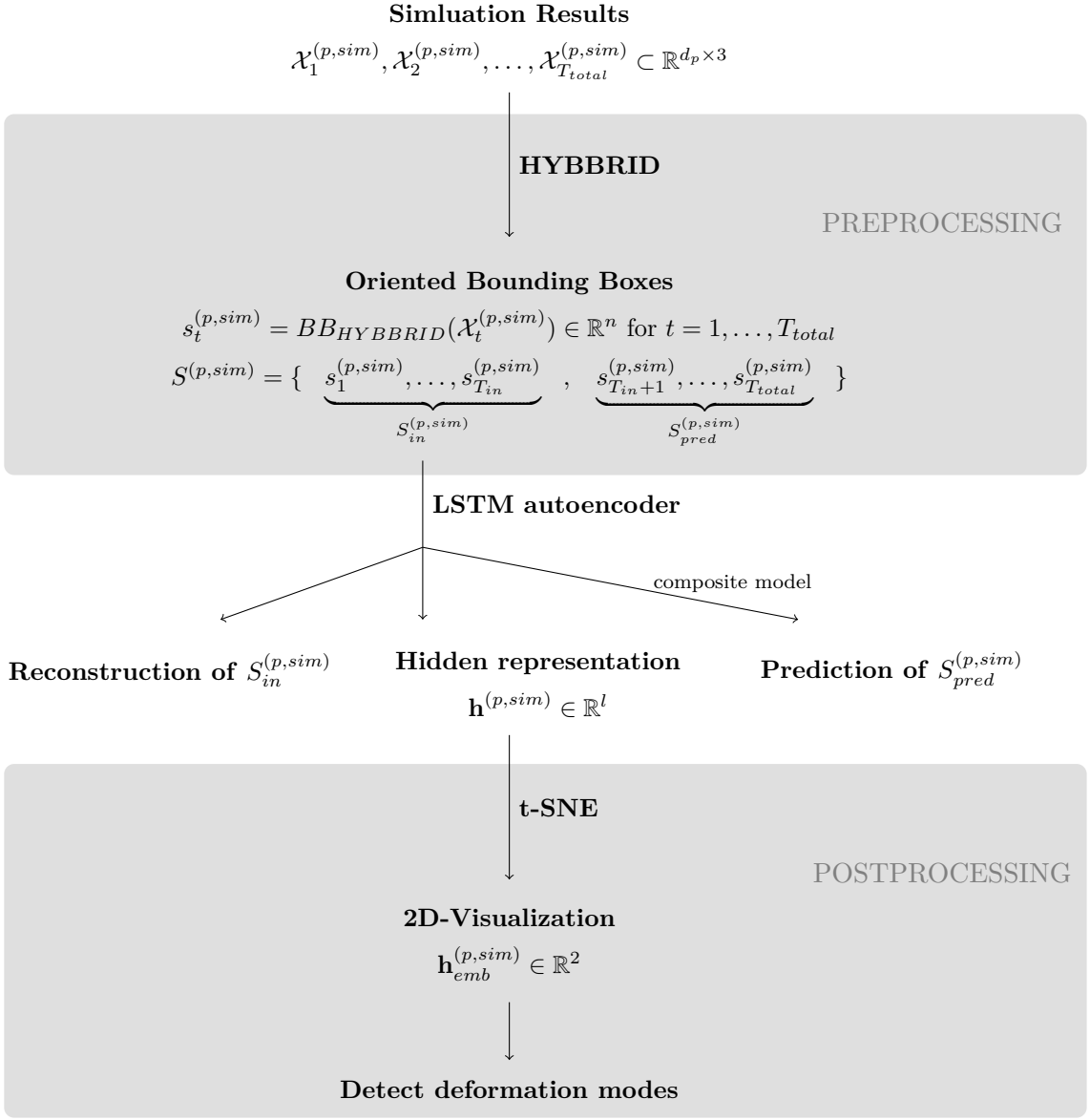


Figure 4.4.: *Input data*: For the chosen car part p, being described by d_p three-dimensional nodes, and the simulation sim select T_{total} timesteps.
Preprocessing: The deforming car part is represented as a sequence $S^{(p,sim)}$ of bounding boxes reducing the dimension to $n = 8 \cdot 3$ for each timestep.
Model: An LSTM autoencoder creates an l-dimensional hidden representation ($l < T_{in} \cdot n$) of the first T_{in} timesteps. Additionally, a reconstruction of the input sequence is created along with a prediction of the future timesteps for the composite model.
Postprocessing: Use t-SNE to visualize the hidden representations for several car parts and simulations and detect deformation modes in the embedding.

5. Training

This short chapter is dedicated to the final preparations of the training and test samples, the description of the training parameters as well as the definition of quality measures. We define quality measures for the reconstructed and predicted boxes. Also we present an accuracy-based score that allows comparison of the hidden representations with respect to their deformation modes detection quality.

5.1. Preparation of Training and Test Samples

The set of simulations SIM is split up into a test and training set, SIM_{test} and SIM_{train} respectively, choosing the first 100 simulations for training and the remaining 96 for testing, so that both branches of bifurcation BI1 are evenly distributed. From the 152 saved timesteps for each simulation, every 5^{th} is selected, leading to time sequences of length $T_{total} = 31$, of which the first T_{in} timesteps are input to the autoencoder. The composite model therefore predicts the last $T_{total} - T_{in}$ timesteps.

The authors of [AIG19] utilize one simulation as one sample and selected only four different parts for the analysis, leading to as many test and training samples as simulations. Although the car parts are approximated by oriented bounding boxes with only 24 dimensions (in comparison to 120 dimensions from spectral coefficients in [AIG19]), a simulation-wise input limits the number of studied car parts. Practical experiments showed that an analysis of more than 10 parts at the same time is not feasible, since the size of the network increases, the memory requirements are not practicable and ultimately, the size of the training set is too small to obtain good results.

To circumvent the limitations on the number of car parts and make the network more compact we utilize a part-wise input with one part as one sample. The autoencoder produces as many hidden representations as simulations for each part, allowing a part-wise analysis. Additionally, the number of training and test samples is now $|SIM \times P|$ instead of $|SIM|$. The part-wise hidden representations are the basis for the localization of deformation modes, since they are evident as different clusters in the low dimensional embedding of an affected part.

Having overcome the limitations on the number of car parts, we select all car parts with more than 100 nodes in the approximating mesh, while excluding round wheel parts, leading to a number of 133 car parts utilized for training.

The part-wise input impedes the autoencoder's ability to directly observe interactions between the parts. Nevertheless, the starting coordinates of the car parts are known and, in addition, the increased number of studied car parts as well as the disposition of part-wise hidden representations outweigh the limitations of a simulation-wise input.

As already commented, the transformation of the oriented bounding boxes is parametrized by translation, rotation and scale. In a first version of the autoencoder the original bounding boxes including all the information are utilized. The mean for each of the three dimensions is normalized to zero and the standard deviation of all the coordinates to one[1]. Therefore, the composite model gives an estimate of the future exact position, including translation, rotation and deformation of the car part.

The car part deformations include a lot of inferred translation, because beams in the front of the car deform in different ways. Therefore, a second version of the LSTM autoencoder was trained utilizing rectified boxes, i.e. the oriented boxes are shifted to the origin and rotated to the standard coordinate axes for all the timesteps. Hence, the rectified boxes solely represent the scale of the car part. They have a mean of zero and are normalized to the standard deviation of one for each dimension[2].

5.2. Training and Model Parameters

Besides deciding on utilizing the original bounding boxes or the rectified boxes, the following model and training parameters are specified.

- An essential parameter is the length T_{in} of the input sequence. If the input sequence is longer, we expect the predictions and deformation mode detection to be better. However, if the input sequence ends before the bifurcations manifest, we cannot expect a meaningful output. Therefore, to detect the bifurcations as soon as possible, T_{in} should denote a timestep after which the bifurcation shortly commences. Generally, the model is trained for $T_{in} = 8, \ldots, 15$, with a more detailed analysis for $T_{in} = 10, 11, 12$, since the bifurcation BI1 starts in timestep 10 (corresponds to timestep 50 of 152 before applying sampling rate 5, see figure A.2).

- The size of the hidden representation is given by the number of hidden neurons l in the encoder. The variable l must be smaller than the size of the input sequence $T_{in} \cdot 24$, because we want the hidden representation to be low-dimensional. At the same time, l should be sufficiently high to distinguish the different deformation modes observed for the 133 parts. As a starting value for all versions we chose 24.

- The number of hidden neurons m in the reconstructing decoder and m' for the predicting decoder is generally higher than the number of neurons l of the encoder.

- If not stated differently, the networks are optimized for 150 epochs, during which every training sample is presented once.

- For the composite model the prediction and reconstruction loss are generally weighted equally with $w_1 = w_2 = 1$, if not stated differently.

[1] Normalization of three dimensional coordinate $x = (x_1, x_2, x_3)$ to $\frac{1}{856.08}(x_1 - 3335.72, x_2 + 3.92, x_3 - 680.19)$

[2] Normalization of three dimensional coordinate $x = (x_1, x_2, x_3)$ to $(\frac{x_1}{349.52}, \frac{x_2}{347.07}, \frac{x_3}{165.35})$

- In case regularization is applied, the weight of the regularizing term $\beta > 0$ is specified.
- The adaptive learning rate optimization algorithm (Adam) [KB14] leads to good convergence behavior of the training and testing errors and is chosen for all optimizations, as in [AIG19] and [SMS15].
- For additional studies of the generalization performance of the architecture, the separation in test and training set as presented in section 5.1, can be varied or a different subset of car parts is presented during the training.

5.3. Quality Measures

The autoencoder yields two different kinds of outputs: the reconstructed and possibly predicted boxes as well as the hidden representation for each part and simulation. The estimated boxes are readily compared to their true values with an error function and, additionally, their orthogonality is measured. For the hidden representation we define an accuracy-based score to evaluate the potential of detecting deformation modes.

5.3.1. Reconstruction and Prediction

First, the reconstructed and predicted boxes are compared to the true values by the mean squared error. The mean squared errors are averaged over all parts and simulations in the training or test set. For more detailed analysis the error is studied for each timestep $t = 1, \dots T_{total}$. To give an estimate of the results' stability, the MSE's mean and standard deviation out of five training runs are given for $T_{in} = 10, 11, 12$. Since there are no similar architectures for the prediction of simulation results, the LSTM autoencoder is compared to the nearest neighbor approximation by the mean squared error and the part-wise standard deviation. For the nearest neighbor approximation, the input sequences from the testing sample are matched to their closest input sequence from the training sample.

Secondly, the boxes are examined with respect to orthogonality. The most common definition of orthogonality for a polygon with four sides or a box is that all angles between two connecting edges have to be right angles. Based on that definition, a measure can be defined by calculating for all 8 vertices of the box the three adjacent angles and average their absolute distances to an orthogonal angle. This approach is costly.

An equivalent condition for a rectangle is that the diagonals bisect each other and are equal in length, which can be easily proven by Thales's theorem. Checking whether a three dimensional polyhedron with six faces, each having four sides, is a rectangular cuboid, is achieved by applying the latter definition for rectangles in three dimensions: A rectangular cuboid's diagonals bisect each other and are equal in length. An equivalent definition is that all faces have to be rectangles.

The orthogonality measure is therefore defined as a combination of comparing the diagonals' lengths and the distances of the diagonals' midpoints. Instead of comparing all

four diagonals to each other we compare each to the average lengths of all four diagonals and the midpoint of the box.

Definition 3 (Orthogonality Measure)**.** *For $i = 1, \dots, 4$ let $diag_i$ be the i^{th} diagonal of the box and m_i^{diag} its midpoint. Additionally let $n^{diag} = \frac{1}{4}\sum_{i=1}^{4} \|diag_i\|^2$ be the average square norm of the diagonals and m^{box} be the midpoint of the box. Define the orthogonality measure of a box as*

$$err_{orth} = \frac{1}{4}\sum_{i=1}^{4} \frac{|\,\|diag_i\|^2 - n^{diag}|}{n^{diag}} + \frac{1}{4}\sum_{i=1}^{4} \frac{|\,\left\|m_i^{diag} - m^{box}\right\|}{n^{diag}}$$

To make the orthogonality measure of different boxes comparable it is normed by the average length of its diagonals.

5.3.2. Hidden Representation

The evaluation and comparison of the hidden representations $\{\mathbf{h}^{(p,sim)} \mid \forall (p, sim) \in P \times SIM\} \subset \mathbb{R}^l$ from different models is challenging, since in the context of unsupervised learning there is not only one correct solution. Additionally, deformation modes are detected visually in a 2D embedding obtained by t-SNE. Since t-SNE is a stochastic algorithm, the results differ slightly from application to application.

There might be more than one correct solution, but since one deformation mode is already known, we concentrate on the ability to detect the known bifurcation BI1 by applying a standard linear classifier. A linear support vector machine (SVM) utilizing l2-penalty and hinge loss is trained for each part $p \in P$ on the embedded 2D-points from the training samples $\{\mathbf{h}_{emb}^{(p,sim)} \mid sim \in SIM_{train}\}$ and as target categories the two bifurcation branches are chosen. The support vector machine tries to find a separating line on the training samples, which we test on the whole set of simulations SIM, illustrated in figure 5.1 for one example part. The score for a fixed part $p \in P$ is the accuracy of the SVM-classifier

$$acc_{\text{SVM}}^{(p)} = \frac{|\{sim \in SIM \mid \mathbf{h}_{emb}^{(p,sim)} \text{ correctly classified}\}|}{n_{simulations}}.$$

Therefore, a high score acc_{SVM} highlights the car parts, that exhibit the studied deformation pattern. Finally, the possible instabilities of the 2D-embeddings obtained by the stochastic t-SNE are compensated by averaging the scores from 5 random applications of the dimension reduction.

Note that, although the SVM could be applied to the l-dimensional hidden representation directly, it is applied to the 2D-embedding obtained by t-SNE. This is motivated by the fact that the user detects deformation modes in the 2D visualization. Because clusters in the embedding are usually sufficiently disentangled, the simple linear classifier yields a quick and adequate separation. In contrast, in l dimensions the clusters often overlap to a higher degree.

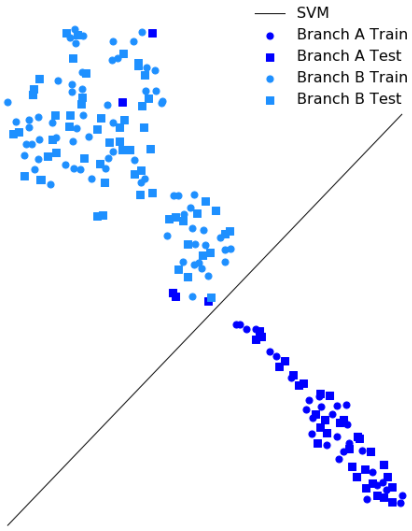


Figure 5.1.: Illustration of the SVM-score acc_{SVM} in the 2D-embedding of the hidden representations $\{\mathbf{h}_{emb}^{(2000005,sim)} \mid sim \in SIM\}$ for part 2000005: $acc_{\text{SVM}}^{(2000005)} = 0.97$. Example with high accuracy taken from composite model analyzing rectified boxes with $T_{in} = 11$.

Finally, this score allows us to compare different models, especially the standard LSTM autoencoder and the composite model with two decoders. We evaluate if the composite model is able to detect the clusters better because it concentrates on representing the future in the hidden representation.

6. Results

This chapter evaluates the different training approaches presented in the previous chapter 5. After comparing the standard and composite model for original boxes as well as a regularization on the orthogonality, the model is applied to rectified boxes.

All the versions are trained for different lengths T_{in} of the input sequence, concentrating around $T_{in} = 10$, when the bifurcation BI1 starts. For selected values, a detailed analysis of the errors and the deformation modes detection is presented. The quality measures are explained in the previous section 5.3.

6.1. Original Bounding Boxes

At first the LSTM autoencoder is applied to original boxes and the significance of the hidden representations for the standard and the composite model is compared.

6.1.1. Standard LSTM Autoencoder

The standard LSTM autoencoder is trained for 150 epochs using $l = 24$ hidden neurons in the encoder and $m = 256$ hidden neurons in the decoder giving a total number of 298,616 trainable parameters. One epoch trains for approximately 16 seconds on a CPU with 16 cores. Table 6.1a lists the layers and how they connect. The network is trained for $T_{in} = 8, \ldots, 15$. For the most interesting timesteps $T_{in} = 10, 11, 12$ the mean and standard deviation of the error from five different trainings are listed in table 6.2a.

Note that for $T_{in} = 10$ the results are unstable and the reconstruction error is higher. It seems that the bifurcation, which starts at timestep 10 is not understood. On the other hand, when using input time sequences with more than 10 timesteps, the errors are stable.

Figure 6.1a shows the SVM-scores for different T_{in} for all parts p with score $acc_{\mathrm{SVM}}^{(p)}$ greater than 0.9 at $T_{in} = 11$. That means 90% of the samples are classified to the correct deformation mode. For some parts the bifurcation is detected, but important parts, for example the left front beam 2000001, are not classified sufficiently well although the bifurcation has already commenced.

6.1.2. Composite Model

The authors of [AIG19] and [SMS15] observed more significant hidden representations when implementing a second decoder, that predicts the future timesteps of the sequence. The version of the autoencoder with two decoders is called composite model, see section 4.3.1

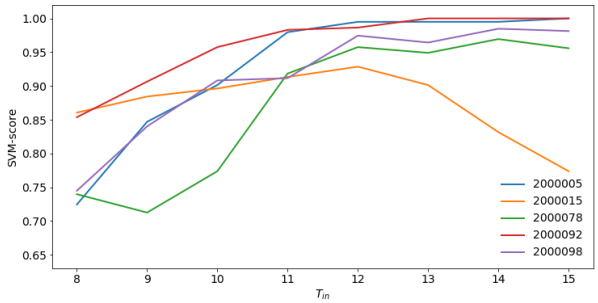
Layer	Output Shape	Param.	Connected to
Input	$(\bullet, T_{in}, 24)$	0	
LSTM 1	$(\bullet, 24)$	4704	Input
Repeat Vector 1	$(\bullet, T_{in}, 24)$	0	LSTM 1
LSTM 2	$(\bullet, T_{in}, 256)$	287744	Repeat Vector 1
Fully Connected	$(\bullet, T_{in}, 24)$	6168	LSTM 2

(a) The standard LSTM autoencoder.

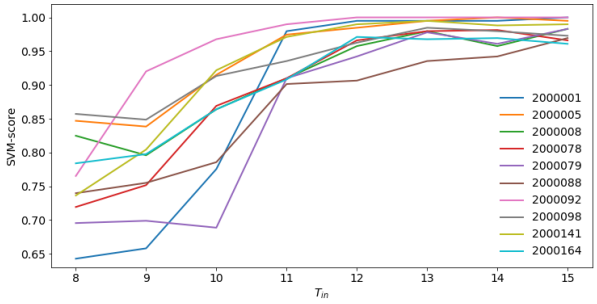
Layer	Output Shape	Param.	Connected to
Input	$(\bullet, T_{in}, 24)$	0	
LSTM 1	$(\bullet, 24)$	4704	Input
Repeat Vector 1	$(\bullet, T_{in}, 24)$	0	LSTM 1
Repeat Vector 2	$(\bullet, T_{total} - T_{in}, 24)$	0	LSTM 1
LSTM 2	$(\bullet, T_{in}, 256)$	287744	Repeat Vector 1
LSTM 3	$(\bullet, T_{total} - T_{in}, 256)$	287744	Repeat Vector 2
Fully Connected 1	$(\bullet, T_{in}, 24)$	6168	LSTM 2
Fully Connected 2	$(\bullet, T_{total} - T_{in}, 24)$	6168	LSTM 3

(b) The composite LSTM autoencoder with a second decoder to predict of the future timesteps.

Table 6.1.: Structure of the LSTM autoencoder in its two versions. The bullets $\bullet$ references the corresponding batch size.



(a) SVM-score for the standard model with one decoder.



(b) SVM-score for the composite model with two decoders.

Figure 6.1.: Plot the SVM-score for different T_{in} and for all parts p, whose score $acc_{\text{SVM}}^{(p)}$ is higher than 0.9 at $T_{in} = 11$. Compare the standard and composite model.

Standard autoencoder: MSE ($\times 10^{-4}$)

	Reconstruction		
T_{in}	Train	Test	Test-STD
10	1.85	1.85	1.029
11	1.40	1.42	0.595
12	1.14	1.15	0.245

(a) Standard autoencoder (one decoder). The presented mean squared errors and standard deviations are obtained by five training runs.

Composite Model: MSE ($\times 10^{-4}$)

	Reconstruction			Prediction			
						Test-STD	
T_{in}	Train	Test	Test-STD	Train	Test	5 Training runs	Part-wise
10	1.07	1.09	0.099	5.70	6.00	0.183	25.56
11	1.11	1.14	0.190	5.31	5.69	0.205	26.67
12	1.10	1.14	0.192	4.73	5.14	0.127	28.05

(b) Composite autoencoder (two decoders). All the presented mean squared errors and standard deviations are obtained by five training runs, except the part-wise STD, which is calculated for the chosen training run.

Nearest Neighbors: MSE ($\times 10^{-4}$)

	Reconstruction	Prediction	
T_{in}	MSE	MSE	Part-wise STD
10	0.24	6.97	32.63
11	0.27	6.49	31.98
12	0.30	6.13	32.85

(c) Nearest Neighbors. Mean squared error of the testing samples and part-wise standard deviation of the prediction.

Table 6.2.: Mean squared errors for the reconstruction and prediction of the original bounding boxes. The training samples consider only the first T_{in} timesteps. Compare the MSE for the standard autoencoder, the composite autoencoder and a nearest neighbor approximation.

The additional branch predicts all the remaining timesteps $T_{total} - T_{in}$ of the simulation. Hence, for $T_{in} = 11$ the composite model reconstructs and predicts all T_{total} timesteps while knowing only the first 35% of the whole sequence. The encoder has the same number of neurons as the standard model, $l = 24$ hidden neurons for the encoder and $m = m' = 256$ hidden neurons for the two decoders. Table 6.1b lists the layers and describes the architecture of the network.

This leads to a total of

$$\underbrace{4704}_{encoder} + 2 \times \underbrace{287744}_{2\ decoders} + 2 \times \underbrace{6168}_{feed-forward} = 592,528$$

trainable parameters. Because the LSTM layers appply parameter sharing over time, the number of parameters does not depend on the length of the input and output sequences.

An encoder with more hidden neurons ($l = 48$) has been trained with similar results, hence, the smaller version is used, whose training takes approximately 36 seconds on a CPU with 16 cores for one epoch.

Reconstruction and Prediction

The average and standard deviation of the mean squared errors of the output are listed in the table 6.2b for the timesteps $T_{in} = 10, 11, 12$ where the bifurcation is starting. The combination of reconstruction and prediction leads to a more stable reconstruction for the critical $T_{in} = 10, 11$ and the MSEs are lower as for the standard model (compare with table 6.2a).

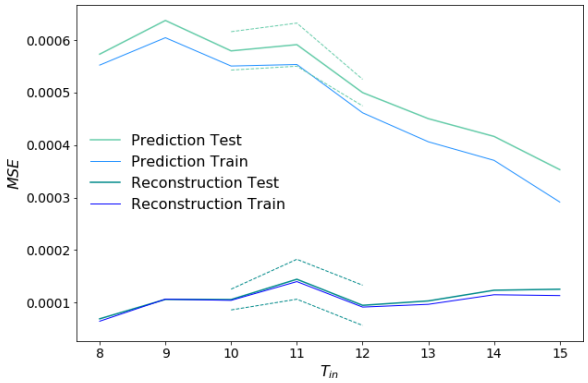
The prediction gives fair approximations, although the mean squared errors of the nearest neighbors can only be improved by approximately 15% and the part-wise standard deviations of the error by approximately 20%, since the dataset has a relatively small variance, see tables 6.2b and 6.2c.

We want to have a more detailed look at the training and testing mean squared errors for different T_{in}, as illustrated in figure 6.2a. For a longer input sequence the predictions improve, as expected. Also, for $T_{in} = 11$, the starting bifurcation leads to slightly higher errors for the prediction and reconstruction branch.

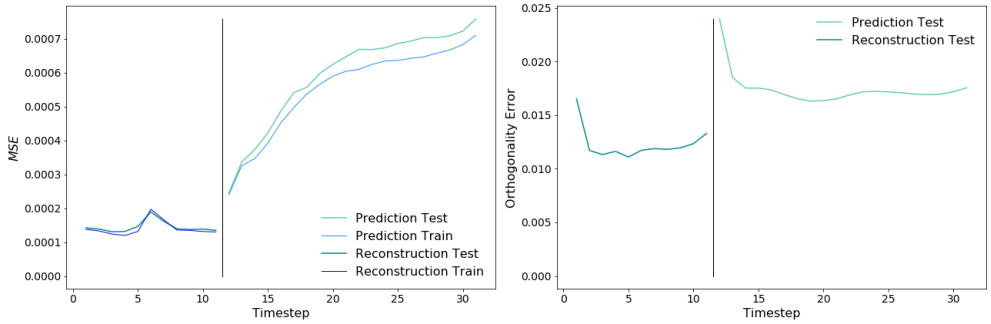
When fixing T_{in} to 11 we can have a detailed look at the mean squared error of the estimated boxes at the different timesteps, figure 6.2b. I want to draw attention to the peak in the error at timestep 6, where the airbag inflates. Also the predictions further in the future are more difficult to make, which leads to a higher MSE.

On the right side of figure 6.2b the orthogonality measure is illustrated for the different timesteps. Interestingly, the LSTM layer improves the orthogonality starting at the second LSTM unit, which indicates that the layers to some extend understand the concept of orthogonal boxes and correct it.

For most of the car parts the estimated boxes have low orthogonality errors, but some predicted boxes show angles of more than 50° away from 90°, especially if their MSE is also high. For that reason regularization on the orthogonality measure has been tested, whose results are presented in section 6.1.3.



(a) MSE of the composite model for all T_{in}. Dotted lines show the $\pm 2\sigma$ interval around the testing mean squared error.



(b) MSE and Orthogonality Error of the composite model for $T_{in} = 11$.

Figure 6.2.: Error measures of the prediction and reconstruction of original boxes with the composite autoencoder.

Deformation Modes Detection

The bifurcation BI1 that is initiated by the left front beam (part id 2000001) is already identified, see section 2.2 and figure A.2. Figure 6.3 shows the 2D-embedding of all the parts that show the same bifurcation at $T_{in} = 11$ and their position in the car. Comparing the illustrations for $T_{in} = 11$ and 12 the propagation of the bifurcation through the car is noticeable. However, most of the detected parts show an inferred bifurcation, since the front beams show different buckling behavior, that influences the translation of parts further in the back.

To localize the bifurcation in the pure deformation without translation and rotation, the rectified bounding boxes have to be analyzed, which is evaluated in the next section. When utilizing rectified boxes we also have a closer look at the detection of deformation patterns in the buckling characteristics of other car components.

6.1.3. Regularization

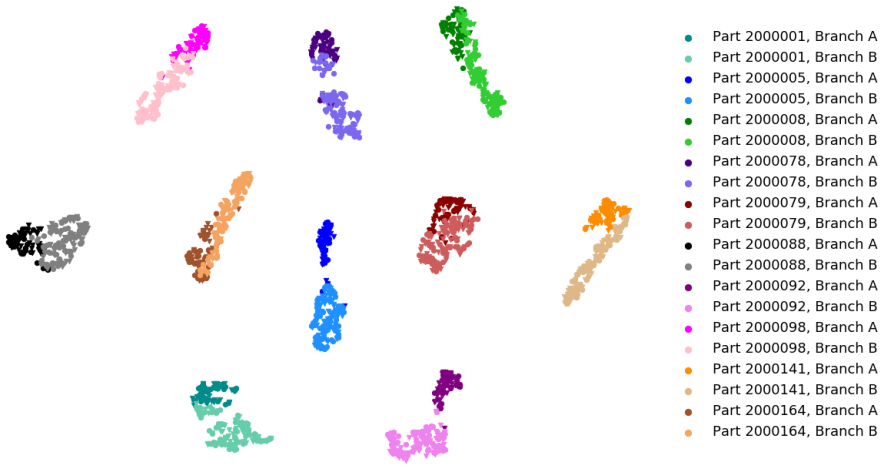
For satisfactory prediction results, the orthogonality of the estimated boxes is essential. For the predicted boxes with the composite model we noticed, that a few boxes are more than 50° away from orthogonality for specific vertices.

The already defined orthogonality measure can be used as a regularization term, which leads to the new loss function

$$L_{reg} = \frac{1}{T_{in}} \sum_{t=1}^{T_{total}} MSE(o_t, s_t) + \beta \ err_{orth,t}$$

for $\beta > 0$, that is optimized instead of the simple mean squared error.

The resulting mean squared errors and orthogonality measures for $\beta = 0.01$ and 0.001 are listed in table 6.3. While the mean squared errors do not change in comparison to the model without regularization (see table 6.2b), the orthogonality slightly improves. The worst angles can be reduces to approximately 40° away from orthogonality. More detailed studies of the orthogonality of the predicted boxes showed that predictions with a low MSE have adequate orthogonality. Only unreliable approximations do not fulfill the orthogonality conditions. Therefore, the regularization on orthogonality has been discarded, since an improvement of the mean squared error should simultaneously solve the orthogonality issues.



(a) 2D-Visualization of hidden representation of parts with high SVM-scores at $T_{in} = 11$.

(b) Illustration of parts with high SVM-score[1] at $T_{in} = 11$.

(c) Illustration of parts with high SVM-score[1] at $T_{in} = 12$.

Figure 6.3.: SVM-score for original boxes utilizing hidden representation of composite model.

[1]Limits for colorcoding: orange for a score >0.9, red for >0.95, bordeaux for >0.98.

T_{in}	MSE ($\times 10^{-4}$)							
	$\beta = 0.01$				$\beta = 0.001$			
	Reconstruction		Prediction		Reconstruction		Prediction	
	Test	Test-STD	Test	Test-STD	Test	Test-STD	Test	Test-STD
10	1.17	0.183	5.87	0.282	1.33	0.378	6.09	0.356
11	1.23	0.296	5.51	0.101	1.09	0.183	5.57	0.158
12	1.36	0.251	5.18	0.148	1.15	0.069	5.22	0.175

(a) Mean squared error under regularization. All the values are averages of five different training runs.

T_{in}	Orthogonality Measure ($\times 10^{-2}$)								
	$\beta = 0.01$			$\beta = 0.001$			$\beta = 0$		
	Total	Rec.	Pred.	Total	Rec.	Pred.	Total	Rec.	Pred.
10	1.41	1.17	1.53	1.55	1.18	1.72	1.67	1.21	1.89
11	1.54	1.26	1.69	1.60	1.25	1.80	1.62	1.23	1.83
12	1.64	1.25	1.88	1.55	1.26	1.73	1.57	1.26	1.76

(b) Orthogonality Measure under regularization. All the values are averages of five different training runs. $\beta = 0$ references the trainings without regularization.

Table 6.3.: Regularization effect on MSE and orthogonality.

6.2. Rectified Bounding Boxes

The composite model in combination with the original bounding boxes is able to detect deformation modes in many parts, although we noticed that it detects mostly the parts, whose translation and rotation are influenced by the bifurcation of other parts. Therefore, many parts with high scores do not show any plastic deformation of their geometry. Hence, the idea evolved to use the rectified bounding boxes, whose center is translated to the origin and which are rotated to the standard coordinate system, as explained in more detail in section 5.1.

The size of the composite model for rectified bounding boxes is the same as for original boxes ($l = 24$ hidden neurons for the encoder and $m = m' = 256$ hidden neurons for the two decoders, leading to 592,528 trainable parameters), because a smaller network with $l = 16, m = 154, m' = 196$ had less significant hidden representations while showing similar MSE.

The training of the chosen network (table 6.1b) has a runtime of 36 seconds for one epoch on a CPU with 16 cores.

Reconstruction and Prediction

The reconstruction and prediction mean squared errors are low and show a similar peak for $T_{in} = 11$ as the composite model applied to original bounding boxes, see figure 6.4c.

When fixing $T_{in} = 11$, we notice in figure 6.4d, that the orthogonality measures are extraordinarily low. Indeed, all the angles in the corners of the estimated boxes are between 87°and 93°.

Additionally, the results are stable, although only slightly better than the nearest

Composite Model: MSE ($\times 10^{-4}$)

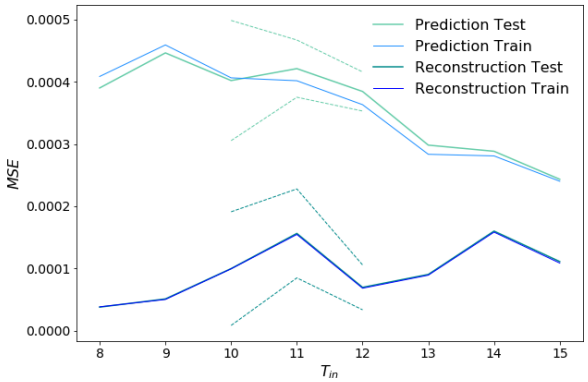
	Reconstruction			Prediction			
T_{in}	Train	Test	Test-STD	Train	Test	Test-STD 5 Training runs	Part-wise
10	1.06	1.07	0.456	4.45	4.43	0.483	15.21
11	0.88	0.90	0.357	3.62	3.82	0.230	15.85
12	0.91	0.93	0.180	3.51	3.66	0.157	17.07

(a) Comparison of the mean squared errors for the reconstruction and prediction branch for different length T_{in} of the input sequence using the composite model with a prediction branch. All the presented mean squared errors and standard deviations are obtained by five training runs, except the part-wise STD, which is calculated for the chosen training run.

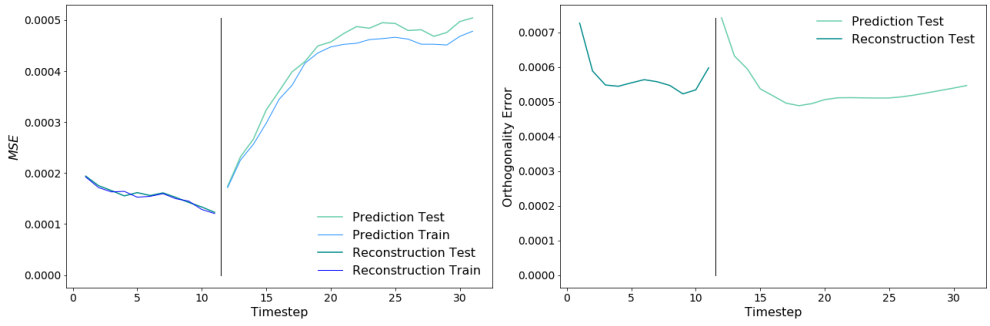
Nearest Neighbors: MSE ($\times 10^{-4}$)

	Reconstruction	Prediction	
T_{in}	MSE	MSE	Part-wise STD
10	0.05	4.43	22.79
11	0.08	4.05	20.78
12	0.10	3.98	22.19

(b) Nearest Neighbors. Mean squared error of the testing samples and part-wise standard deviation of the prediction.



(c) MSE for all T_{in}. Dotted lines show the $\pm 2\sigma$ interval around the testing mean squared error.



(d) MSE and Orthogonality for $T_{in} = 11$.

Figure 6.4.: Prediction and reconstruction of rectified boxes utilizing a composite model.

neighbor results (for $T_{in} = 11$ the composite model has 5% lower MSE, 25% lower part-wise standard deviation), which are listed in figure 6.4b.

Deformation Modes Detection

An SVM applied to the 2D embeddings of the low dimensional representation detects the deformation modes. We interpret car part p as deforming significantly in the deformation modes if its SVM-score $acc_{\text{SVM}}^{(p)}$ is higher than 0.9. That means given the linear SVM on the training data, more than 90% are classified correctly with respect to the branches of the bifurcation.

Figure 6.5 presents the two-dimensional embedding obtained by t-SNE. We detect the bifurcation in less parts than with the original boxes, which is to be expected, since the rectified boxes contain less information, therefore less differences can be detected. Nevertheless, a part with high score experiences changes in scale, which indicates plastic deformation. Parts that are assigned to deformation modes due to their indirect rotation and translation are not as inclined to fracture or strain. Regarding incorrectly classified samples in figure 6.5b, we notice that they are generally located close to the correct cluster.

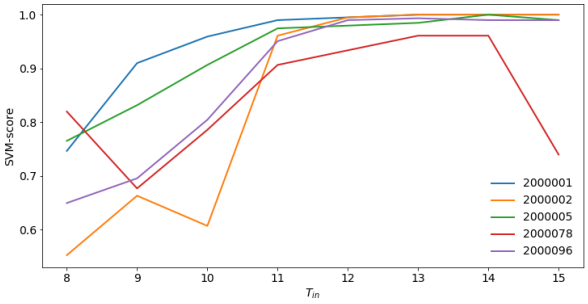
The parts which are detected with high significance are mainly beams on the left side of the car. As observed in figure 6.6, the bifurcation BI1 can be detected in the left front beam 2000001 with 95% accuracy at $T_{in} = 10$, which is right upon manifestation.

Some car parts show clusterings different to the bifurcation BI1 in the left front beam. The right front beams (part 2000003 and 2000004) show a similar bifurcation in the deformation as the left front beams, although the smaller branch includes only 10% of the simulations. Figure 6.7 visualizes the 2D embedding and illustrates the car parts, which significantly show the newly detected bifurcation BI2. The detection is less stable and only possible for a length of the input sequence of $T_{in} \geq 12$, since the bifurcation is unbalanced with respect to the quantity. Figure A.5 in the appendix illustrates the two branches of the bifurcation. The limits for significance are set higher at an SVM-score of 92%, since the small branch only includes 10% of the samples.

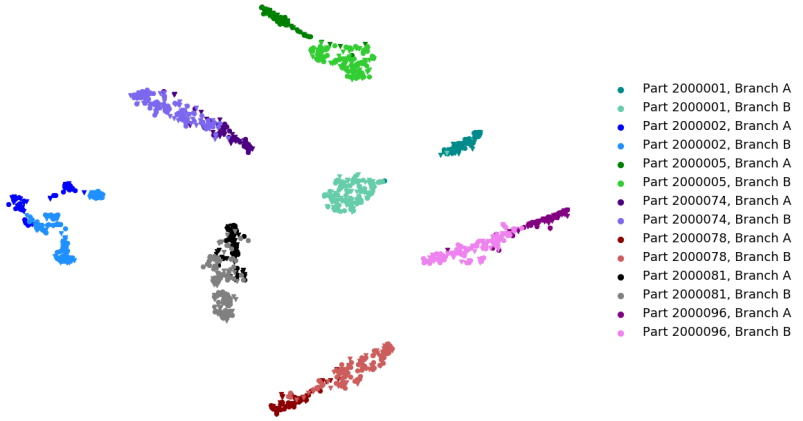
Another clustering proved to be significant with respect to the SVM score in many car parts in the right front of the car, especially in the parts surrounding the wheel. Figure 6.8 visualizes the 2D embedding and illustrates the car parts, which show an arrangement corresponding to the newly detected bifurcation BI3. It seems to be initiated by car part 2000049. Figure A.6 in the appendix illustrate the two branches of the bifurcation.

6.2.1. Unequal Weighting of Loss Functions

To improve the quality of the prediction of the composite model we try unequal weighting of the losses. The loss of the prediction is weighted by a $w_2 > 1$ while the loss of the reconstruction is weighted by $w_1 = 1$ as before, giving the prediction more importance during the training. However the unequal weighting of the losses has no improving effect on the prediction's quality, but leads to an increased MSE of the reconstruction, as displayed in table 6.4.



(a) Plot the SVM-score over time T_{in} for all parts p, whose score $acc_{\mathrm{SVM}}^{(p)}$ is greater than 0.9 at $T_{in} = 11$



(b) 2D-Visualization of hidden representation of parts with high SVM-scores at $T_{in} = 11$.

Figure 6.5.: SVM-scores for bifurcation BI1 over time and 2D-embedding for selected car parts represented by rectified boxes, which are analyzed by the composite model.

T_{in}	MSE ($\times 10^{-4}$)							
	$w_1 = 1$, $w_2 = 5$				$w_1 = 1$, $w_2 = 10$			
	Reconstruction		Prediction		Reconstruction		Prediction	
	Test	Test-STD	Test	Test-STD	Test	Test-STD	Test	Test-STD
10	1.10	0.136	4.26	0.262	1.28	0.306	4.86	0.432
11	1.23	0.352	4.22	0.508	1.73	1.360	3.82	0.359
12	1.42	0.130	3.60	0.412	1.10	0.352	3.54	0.223

Table 6.4.: Effects of unequal weighting of losses on the MSE of the test set. All the values are averages of five different training runs.

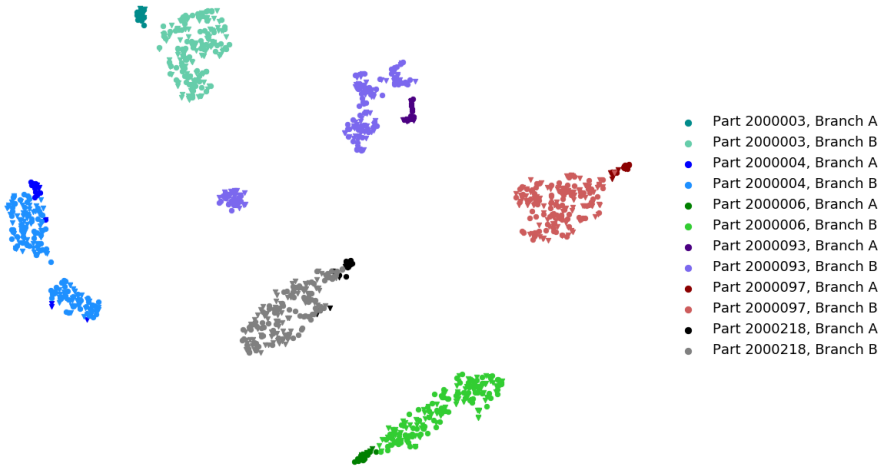
(a) $T_{in} = 10$.

(b) $T_{in} = 11$.

(c) $T_{in} = 12$.

Figure 6.6.: Illustration of parts in which the deformation mode BI1 is detected with a high SVM-score[1]. Detection of deformation modes in the low dimensional embeddings of a composite model based on input sequences of rectified boxes of length $T_{in} = 10, 11, 12$.

[1]Limits for colorcoding: orange for an SVM-score >0.9, red for >0.95, bordeaux for >0.98.



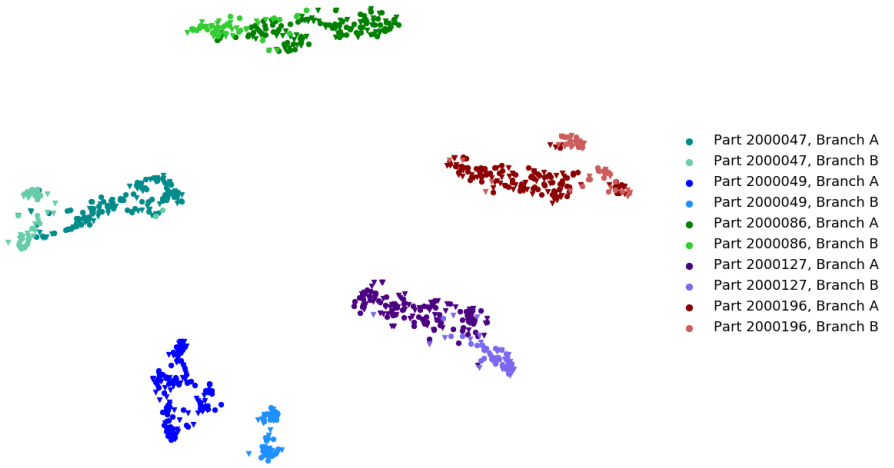
(a) 2D-Visualization of hidden representation of parts with high SVM-scores at $T_{in} = 12$.

(b) Illustration of parts with high SVM-score[1] at $T_{in} = 11$.

(c) Illustration of parts with high SVM-score[1] at $T_{in} = 12$.

Figure 6.7.: Visualization of SVM-score for bifurcation BI2 that has been detected with the composite model using rectified boxes. Highlight selected parts, where the new bifurcation can be observed.

[1]Limits for colorcoding: orange for a score >0.92, red for >0.95, bordeaux for >0.98.



(a) 2D-Visualization of hidden representation of parts with high SVM-scores at $T_{in} = 11$.

(b) Illustration of parts with high SVM-score[1] at $T_{in} = 10$.

(c) Illustration of parts with high SVM-score[1] at $T_{in} = 11$.

Figure 6.8.: Visualization of SVM-score for bifurcation BI3 that has been detected with the composite model using rectified boxes. Highlight selected parts, where the new bifurcation can be observed.

[1]Limits for colorcoding: yellow for a score >0.85, orange for >0.9, red for >0.95.

6.2.2. Generalization Performance of the Architecture

The generalization performance of an architecture describes its ability to understand previously unseen samples. We want to test the generalization performance of the LSTM autoencoder by varying the training data. We try to use only a subset of all the selected 133 parts. In a second experiment, the network is trained with the simulations from only one branch of the bifurcation BI1.

Selection of Training Samples

The quantity of training samples, which in our case depends on the number of analyzed car components, determines mainly the training time for one epoch. Additionally, a larger training set requires more epochs. Therefore, a relevant question is whether the selection of all car parts for training is necessary for the presented results.

With less than 50 randomly selected parts, the results are not satisfactory, because the output of neural networks generally depends strongly on the variance in the training set. Even when randomly selecting 100 out of 133 parts, the average mean squared error of all parts increases. Interestingly, for an unseen beam the predictions are comparable to before, since other parts of similar shape are contained in the data set. If parts of similar shape are not included in the training set, predictions are inaccurate, which is why the mean squared error increases, see table 6.5.

For better results, the parts have to be selected with respect to a the possible variations of form, which is a time-consuming task and requires detailed knowledge of the car model.

Therefore, the selection of all parts is recommendable, because a preselection of relevant parts is challenging and the runtime for all parts is reasonable. As long as similar parts exist in the sample, the predictions regarding omitted ones are still reliable.

Composite Model: MSE ($\times 10^{-4}$)

	Reconstruction		Prediction	
T_{in}	Train	Test	Train	Test
10	0.79	14.78	4.73	36.27
11	0.67	23.19	3.51	33.46
12	1.49	22.38	2.98	21.99
13	1.28	37.64	3.13	12.95

Table 6.5.: Mean squared errors of the composite model for different T_{in} utilizing only 100 out of 133 parts for training. Testing errors is calculated for all 133 parts.

Detection of Unknown Deformation Modes

We want to test the generalization power of the LSTM autoencoder for unseen deformation. For that purpose we consider a training set, that includes only those simulations, that belong to branch B of the bifurcation BI1 iniciated by the left front beam. The bigger branch B with 129 simulations is utilized for the training, whereas the remaining 67

simulations are used for testing. The testing mean squared error is notably higher for prediction as well as reconstruction and the training errors are comparable to the standard model, including both branches in the training data, see figure 6.9a.

Although the quality of the prediction is lower for the unseen branch, it is comparable for the deformation mode detection in the 2D-embedding. Figure 6.9 shows the embedding and the SVM-scores for significant parts. For the training of the SVM the full training and test data have to be used, because samples of both deformation modes are necessary. Additionally, since the autoencoder does not know about the strong bifurcation, it detects another buckling behavior for part 2000002, which is the inner side of the left front beam.

6.3. Comparison of Model Versions

In summary, the versions of the LSTM autoencoder are listed along their general training success on the TRUCK data set and an evaluation of their advantages and limitations:

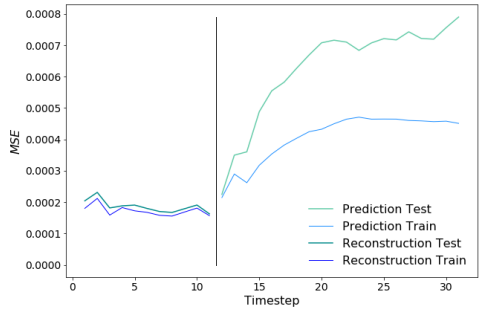
Composite vs. Standard Model The composite model with two decoders, one for reconstruction and an additional decoder for prediction as presented in [SMS15], yields more significant low dimensional embeddings. The prediction module leads to more relevant embeddings regarding the deformation in future timesteps. Additionally, the composite model yields an approximate prediction of the bounding box positions that has about 15% lower MSE than the nearest-neighbor-based approximation when utilizing original bounding boxes.

Original Bounding Boxes The LSTM autoencoder, when applied to original bounding boxes representing translation, rotation and scale, successfully detects the deformation modes in the car parts' low dimensional representations, although many detected patterns are solely induced by part interaction and indirect translation differences without any plastic deformation. The predictions of the boxes' position based only on the low dimensional embeddings yield satisfying results that can be used as an in-situ, preliminary estimate of the final simulation result.

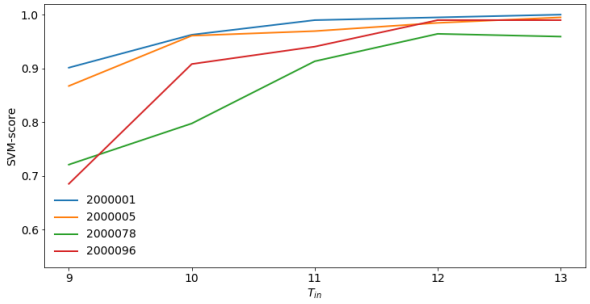
Regularization on Orthogonality If the prediction quality of the oriented bounding boxes is poor, the output boxes are not rectangular, deviating in excess of 40° from a right angle. A regularization on orthogonality does not improve the quality nor the orthogonality of the predicitions significantly.

Rectified Bounding Boxes When utilizing only the scale of the oriented bounding boxes as the input, passive translation-based deformation modes in parts are excluded and the LSTM autoencoder focuses on plastic deformation in the car parts. The deformation pattern BI1 is detected in less car parts, but with higher significance sooner in the simulation. Additionaly, two new deformation modes BI2 and BI3 are detected by a part-wise analysis of the low dimensional embeddings.

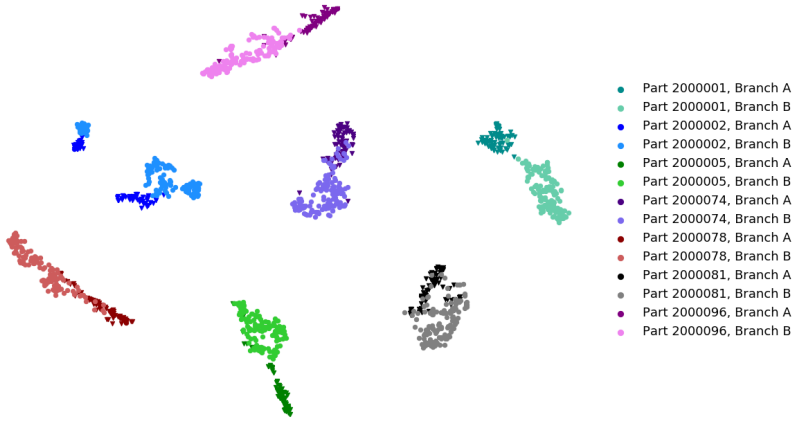
Generalization Performance As for all neural networks, the reliability of the LSTM autoencoder depends highly on a representative distribution of the training



(a) MSE and Orthogonality for $T_{in} = 11$.



(b) Plot the SVM-score for different T_{in} and for all parts, whose score is bigger than 0.9 at $T_{in} = 11$



(c) 2D-Visualization of hidden representation of parts with high SVM-scores at $T_{in} = 11$.

(d) Illustration of parts with high SVM-score[1] at $T_{in} = 11$.

Figure 6.9.: Error and SVM-score of the composite model applied to rectified boxes while utilizing only branch B of bifurcation BI1 for training.

[1]Limits for colorcoding: orange for a score >0.9, red for >0.95, bordeaux for >0.98.

set. Therefore, an incomplete training set with omitted parts is not sufficient for adequate training results. In contrast, if not all the deformation patterns are included in the training set, the LSTM autoencoder learns the unseen patterns in the low dimensional representations.

7. Conclusion and Outlook

Conclusion

We modified and evaluated an LSTM autoencoder [SMS15] for in-situ analysis and prediction of car crash simulations, utilizing a preprocessing step converting 3D surface meshes to oriented bounding boxes. Different algorithms for calculating bounding boxes were investigated theoretically as well as practically and the HYBBRID algorithm [CGM11] adapted to the specific properties of simulation data. The adaptions reduce the runtime of the original HYBBRID algorithm by a factor of three while the resulting bounding boxes are stable and their volumes similar to those of optimal bounding boxes. They distinguish effectively between different deformation patterns.

The evaluation of different versions of the LSTM autoencoder demonstrate that the composite model is more suitable for in-situ analysis of simulation data. The use of a prediction decoder leads to more future relevant hidden representations and the part-wise detection of deformation modes in the deformation behavior is possible sooner. The hidden representations simplify the detection of deformation modes, since they are detectable in 2D visualizations of the feature representations. The developed accuracy-based SVM score enables identification of parts that are effected by the studied deformation mode. Another advantage are the predictions, giving a first estimate of the bounding box positions in future timesteps, complementing the detailed simulations results.

Depending on the goal of the analysis, we recommend the use of either rectified bounding boxes or original bounding boxes. Rectified bounding boxes are shifted to the origin and rotated to the standard coordinate axes. It is not necessary to estimate the rigid motion by follow-up points, since that information is computed during the calculation of oriented bounding boxes. Rectified bounding boxes concentrate solely on differences in the buckling behavior and allow the in-situ detection of deformation modes in critical parts that are prone to fraction or strain. The LSTM autoencoder in combination with original bounding boxes allows highlighting all parts that the deformation mode affects, even though only indirectly by translation and rotation. The predicted bounding boxes have a 15% lower MSE than nearest neighbor approximations. The predictions' quality depends on a representative distribution of the training samples. However, the detection of deformation modes in the hidden representation is reliable and stable even for omitted patterns of deformation.

Outlook

Due to promising results for the TRUCK dataset, the LSTM autencoder in combination with oriented bounding boxes should be investigated for further data sets. Of particular interest is the use of the trained weights as a pretrained model for more complex data sets. The parameter sharing over time makes the model architecture independent of the number of analyzed timesteps so it can be directly applied to any simulation results preprocessed by oriented bounding boxes.

The architecture offers more possibilities to improve the prediction results. The LSTM autoencoder, which we have evaluated, considered the input and output time sequences separately. For training, the consideration of the future timesteps in the input sequence might improve the prediction result [AIG19]. In addition, the investigation of the provision of additional information, including model parameters such as material characteristics or classes of car parts to the decoders is noteworthy. In this way, the prediction quality can improve.

There are deformation modes indistinguishable by bounding boxes. Therefore, the study of other shape representations that are able to capture the deformation patterns and handle geometry changes might be worthwhile. How to study part interaction remains an open question.

Altogether, I consider the potential of the LSTM autoencoder in combination with oriented bounding boxes or other shape representations to be considerable. I am looking forward to further developments which the future will bring and hope to be able to contribute to them.

A. Appendix

A.1. Proof of Bounds on the Volume of PCA-Based Bounding Boxes

This section contains sketches to the proofs of the lower bounds on $\kappa_{d,i}$ for $d = 2$ and $d = 3$ as shown in [Dim+09]. They figuratively illustrate the limitations of the PCA-based bounding boxes.

Lemma 4. *$\kappa_{2,1} \geq 2$ and $\kappa_{2,2} \geq 2$.*

Proof. The lower bounds of the approximation factors $\kappa_{2,1}$ and $\kappa_{2,2}$ are shown using a rhombus with side length 1 as in figure 3.5. As long as the angles in the corners are not equal to 90° there are two different principal components, which are at the same time the rhombus' diagonals. When one angle in a corner approaches 90° the area of the optimal bounding box approaches 1, but the area of the BB_{PCA} approaches $\sqrt{2} \times \sqrt{2} = 2$. □

Lemma 5. *$\kappa_{3,2} \geq 4$ and $\kappa_{3,3} \geq 4$.*

Proof. Using the same idea as in the proof for the approximation factor κ in 2D, we consider now a dipyramid, having a rhombus with side length $\sqrt{2}$ as its base. The other sides have length $\frac{\sqrt{3}}{2}$. The dipyramid with its principal components and the two bounding boxes BB_{PCA} and BB_{opt} are illustrated in figure A.1. When the angles of the rhombus at the dipyramids base approach 90°, the volume of the PCA-based bounding box is $2 \times 2 \times \sqrt{2}$, whereas the optimal bounding box has the volume $1 \times 1 \times \sqrt{2}$. Therefore, the approximation factors $\kappa_{3,2}$ and $\kappa_{3,3}$ are at least 4. □

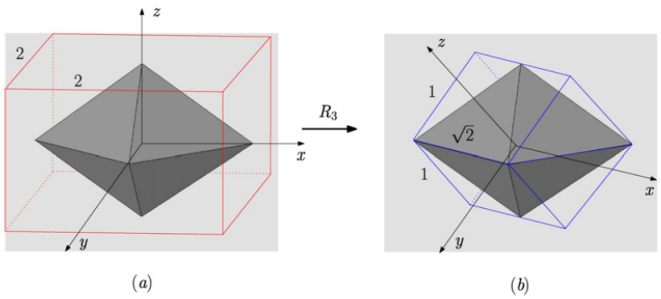


Figure A.1.: Example showing the lower bound for the area of BB_{PCA} in $\mathbb{R}^3$. Illustration of the bounding boxes (a) BB_{PCA} and (b) BB_{opt}, from [Dim+09].

The proofs show, that PCA-based bounding boxes tend to fail, when the eigenvalues are of similar value and the side lengths of the optimal bounding boxes are almost equal to each other. In that case there is no strong principal direction, since the point set $\mathcal{X}$ is distributed similarly in all directions.

A.2. Illustrations

This chapter contains the illustration of the deformation modes of different parts and gives explanatory plots to the oriented bounding boxes.

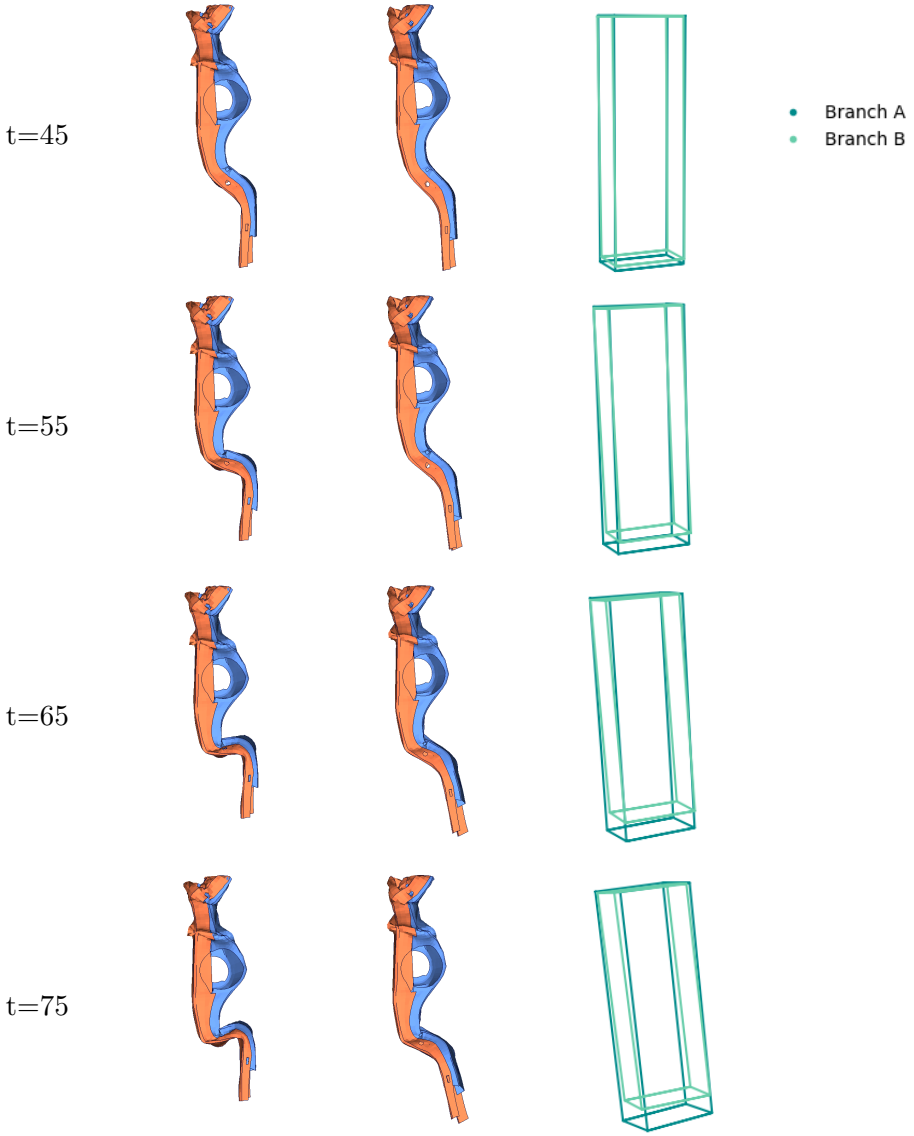


Figure A.2.: Bifurcation BI1: Vizualization of left front beams (Part 2000001 in orange, Part 2000002 in blue) in its two different deformation modi. The first column shows deformation mode A, the second column shows deformation mode B. The third column illustrates the corresponding bounding boxes. Each row is a different timestep.

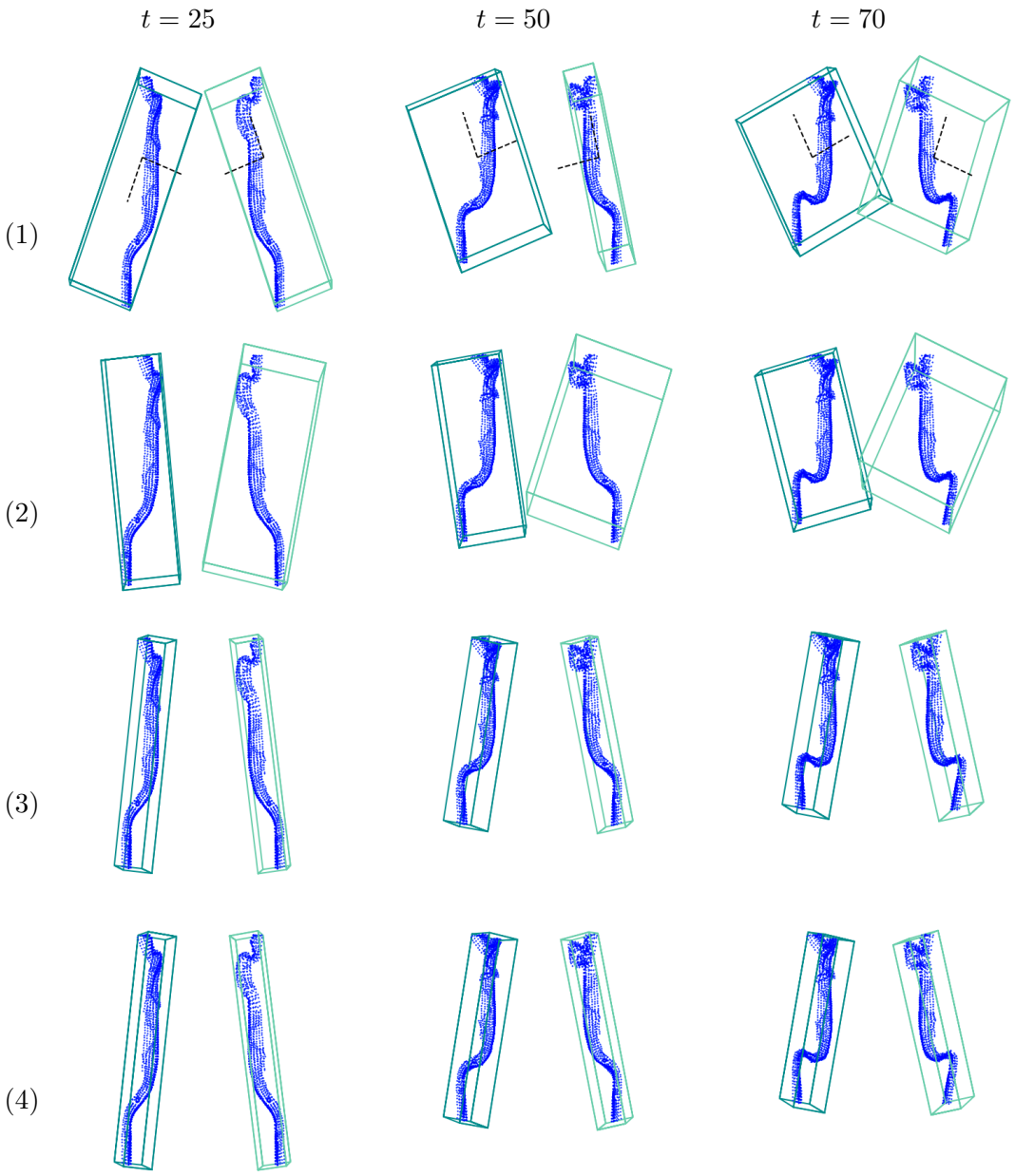


Figure A.3.: Demonstration of resulting bounding boxes for the front beams 2000001 and 2000003 using the algorithms: (1) PCA-based on points with eigenvectors, (2) PCA-based on triangulation of convex hull (3) HYBBRID algorithm and (4) exact solution. Observe that the results from the PCA-based algorithm are not stable when comparing different timesteps.

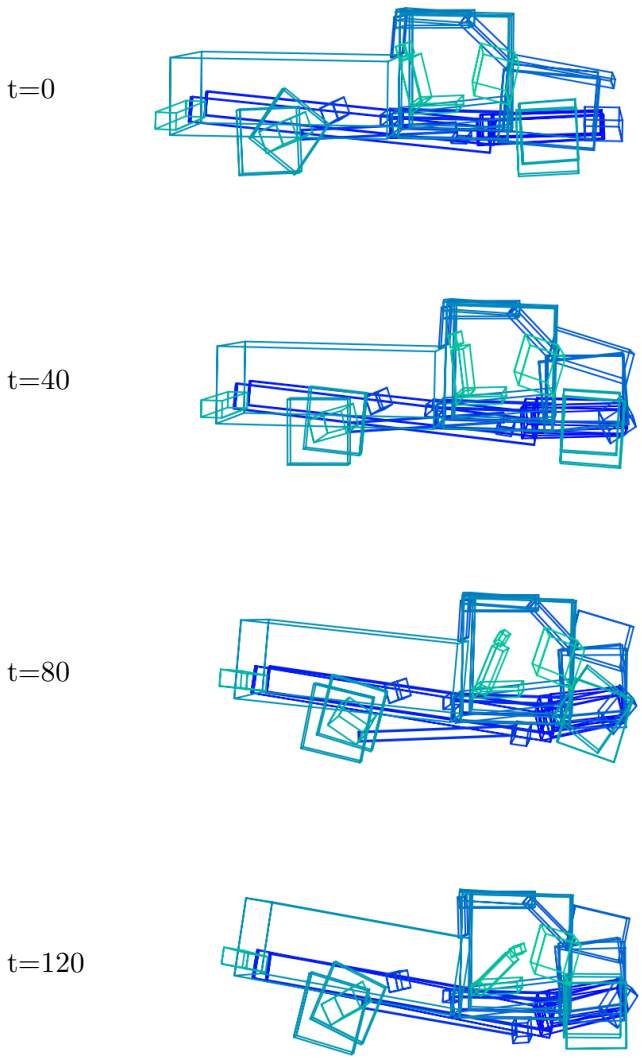


Figure A.4.: Illustration of the preprocessing result. Approximate the finite element mesh by optimal bounding boxes for each model part. Showing selected parts and timesteps from simulation 001

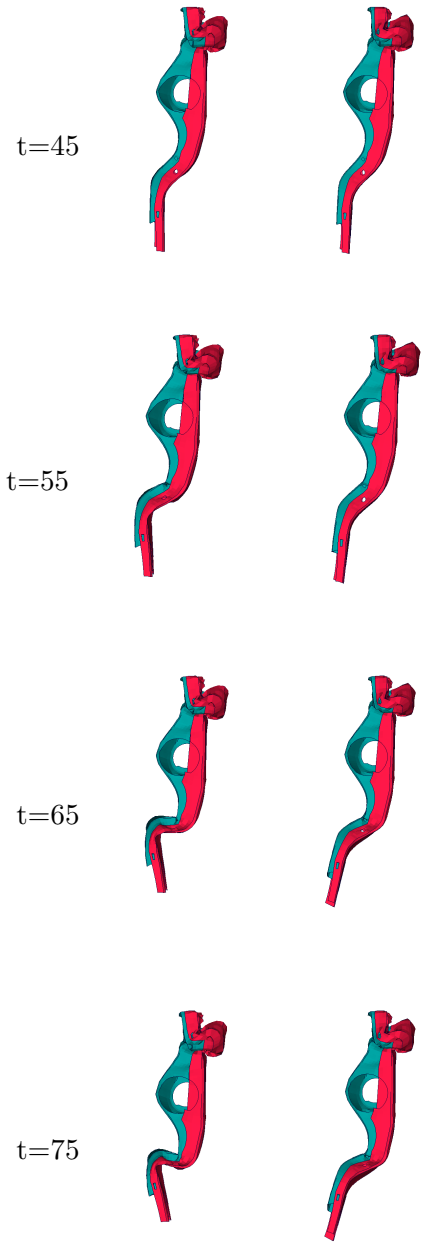


Figure A.5.: Bifurcation BI2: Vizualization of left front beams (Part 2000003 in red, Part 2000004 in green) in its two different deformation modi. The first column shows the beams from simulation 001 which belongs to the bigger branch, the second column shows beams from simulation 018 which belongs to the small branch. Each row is a different timestep.

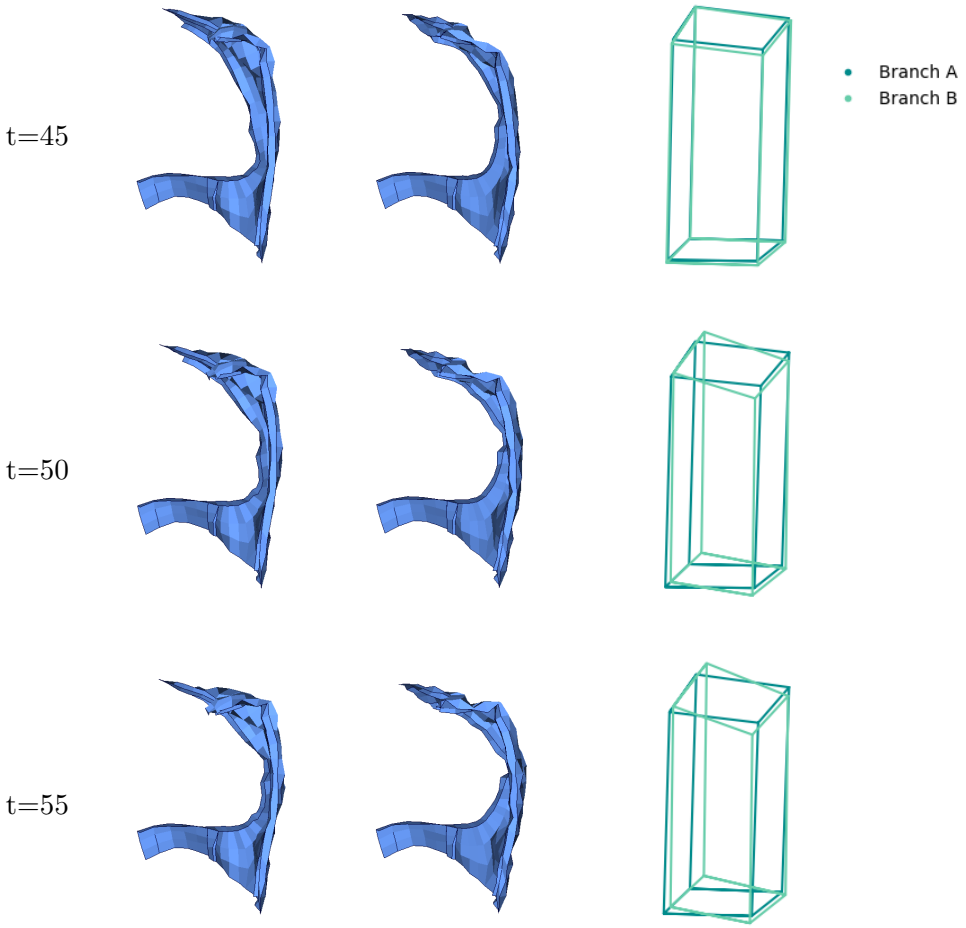


Figure A.6.: Bifurcation BI3: Vizualization of Part 2000049 in the right front of the car in two different deformation modi. The first column shows the beams from simulation 001 which belongs to branch A, the second column shows beams from simulation 015 which belongs to branch B. The third column illustrates the corresponding bounding boxes. Each row is a different timestep.

Bibliography

[AIG19] Amin Abbasloo, Rodrigo Iza-Teran, and Jochen Garcke. "Unsupervised Learning of Automotive 3D Crash Simulations using LSTMs". In: *ICLR 2020 (under review as a conference paper)*. 2019.

[Bar+96] Gill Barequet, Bernard Chazelle, Leonidas J Guibas, Joseph SB Mitchell, and Ayellet Tal. "BOXTREE: A hierarchical representation for surfaces in 3D". In: *Computer Graphics Forum*. Vol. 15. 3. Wiley Online Library. 1996, pp. 387–396.

[BGG16] Bastian Bohn, Jochen Garcke, and Michael Griebel. "A sparse grid based method for generative dimensionality reduction of high-dimensional data". In: *Journal of Computational Physics* 309 (2016), pp. 1–17.

[Boh+13] Bastian Bohn, Jochen Garcke, Rodrigo Iza-Teran, Alexander Paprotny, Benjamin Peherstorfer, Ulf Schepsmeier, and Clemens August Thole. "Analysis of car crash simulation data with nonlinear machine learning methods". In: *Procedia Computer Science* 18 (2013), pp. 621–630.

[Boi+04] Paul Du Bois et al. *Vehicle Crashworthiness and Occupant Protection. The Science of Microfabrication*. Southfield, Michigan, USA: American Iron and Steel Institute, 2004.

[Bro+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[CGM11] Chia-Tche Chang, Bastien Gorissen, and Samuel Melchior. "Fast Oriented Bounding Box Optimization on the Rotation Group $SO(3, \mathbb{R})$". In: *ACM Transactions on Graphics (TOG)* 30.5 (2011), p. 122.

[Cho+15] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[Dim+09] Darko Dimitrov, Christian Knauer, Klaus Kriegel, and Günter Rote. "Bounds on the quality of the PCA bounding boxes". In: *Computational Geometry* 42.8 (2009), pp. 772–789.

[Eur18] Directorate General for Transport European Commission. *European Commission, Vehicle Safety*. `https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/ersosynthesis2018-vehiclesafety.pdf`. Feb. 2018.

[FS75] Herbert Freeman and Ruth Shapira. "Determining the minimum-area encasing rectangle for an arbitrary closed curve". In: *Communications of the ACM* 18.7 (1975), pp. 409–413.

[Gao+19] Lin Gao, Yu-Kun Lai, Jie Yang, Zhang Ling-Xiao, Shihong Xia, and Leif Kobbelt. "Sparse data driven mesh deformation". In: *IEEE transactions on visualization and computer graphics* (2019), pp. 1–1.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. "OBBTree: A Hierarchical Structure for Rapid Interference Detection". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 171–180.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. first edition. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[Gra13] Alex Graves. "Generating Sequences With Recurrent Neural Networks". In: *CoRR* abs/1308.0850 (2013). arXiv: 1308.0850.

[GS14] Lars Graening and Bernhard Sendhoff. "Shape mining: A holistic data mining approach for engineering design". In: *Advanced Engineering Informatics* 28.2 (2014), pp. 166–185.

[Gue+18] Y. Le Guennec, J.-P. Brunet, F.-Z. Daim, M. Chau, and Y. Tourbier. "A parametric and non-intrusive reduced order model of car crash simulation". In: *Computer Methods in Applied Mechanics and Engineering* 338 (2018), pp. 186–207.

[Hoc+01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[Hoc91] Sepp Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". In: *Diploma, Technische Universität München* 91.1 (1991).

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems*. second edition, 1992. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780.

[IG19] Rodrigo Iza-Teran and Jochen Garcke. "A Geometrical Method for Low-Dimensional Representations of Simulations". In: *SIAM/ASA Journal on Uncertainty Quantification* 7.2 (2019), pp. 472–496.

[Jyl15] Jukka Jylänki. "An exact algorithm for finding minimum oriented bounding boxes". In: *Semantic Scholar. Available online: https://pdfs.semanticscholar.org/a76f/7da5f8bae7b1fb4e85a65bd3812920c6d142.pdf* (2015). Accessed: 2010-09-12.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[LeC+89] Yann LeCun et al. “Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning”. In: *IEEE Communications Magazine* 27.11 (1989), pp. 41–46.

[Lit+17] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. “Deep functional maps: Structured prediction for dense shape correspondence”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5659–5667.

[Mas+15] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. “Geodesic convolutional neural networks on riemannian manifolds”. In: *Proceedings of the IEEE international conference on computer vision workshops*. 2015, pp. 37–45.

[McK98] Ken IM McKinnon. “Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point”. In: *SIAM Journal on Optimization* 9.1 (1998), pp. 148–158.

[Mey07] Martin Meywerk. *CAE-Methoden in der Fahrzeugtechnik*. Springer, 2007.

[MH08] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[Mon+17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. “Geometric deep learning on graphs and manifolds using mixture model CNNs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5115–5124.

[NM65] John A Nelder and Roger Mead. “A simplex method for function minimization”. In: *The Computer Journal* 7.4 (1965), pp. 308–313.

[ORo85] Joseph O’Rourke. “Finding minimal enclosing boxes”. In: *International Journal of Computer & Information Sciences* 14.3 (1985), pp. 183–199.

[ORo98] Joseph O’Rourke. *Computational Geometry in C*. second edition. New York, NY, USA: Cambridge University Press, 1998.

[Qia+18] Yi-Ling Qiao, Lin Gao, Yu-Kun Lai, and Shihong Xia. “Learning Bidirectional LSTM Networks for Synthesizing 3D Mesh Animation Sequences”. In: *arXiv preprint arXiv:1810.02042* (2018).

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[SMS15] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. “Unsupervised Learning of Video Representations using LSTMs”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Lille, France, 2015, pp. 843–852.

[SSW17] Tobias Sprügel, Tina Schröppel, and Sandro Wartzack. “Generic approach to plausibility checks for structural mechanics with deep learning”. In: *DS 87-1 Proceedings of the 21st International Conference on Engineering Design (ICED17)*. Vol. 1: Resource-Sensitive Design, Design Research Applications and Case Studies. Vancouver, Canada, 2017, pp. 299–308.

[SW94] Inge Söderkvist and Per-Åke Wedin. “On condition numbers and algorithms for determining a rigid body movement”. In: *BIT Numerical Mathematics* 34.3 (1994), pp. 424–436.

[Tan+18a] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. “Variational autoencoders for deforming 3d mesh models”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5841–5850.

[Tan+18b] Qingyang Tan, Lin Gao, Yu-Kun Lai, Jie Yang, and Shihong Xia. “Mesh-based autoencoders for localized deformation component analysis”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[Tou83] Godfried Toussaint. “Solving geometric problems with the rotating calipers”. In: *Proc. IEEE Melecon*. Vol. 83. 1983, A10.

[VSR01] D. V. Vranic, D. Saupe, and J. Richter. “Tools for 3D-object retrieval: Karhunen-Loeve transform and spherical harmonics”. In: *2001 IEEE Fourth Workshop on Multimedia Signal Processing (Cat. No.01TH8564)*. 2001, pp. 293–298.

[Zha+10] Zhijie Zhao, Jin Xianlong, Yuan Cao, and Jianwei Wang. “Data mining application on crash simulation data of occupant restraint system”. In: *Expert Systems with Applications* 37 (8 2010), pp. 5788–5794.

Statement of authorship

I hereby confirm that the work presented in this master thesis has been performed and interpreted solely by myself except where explicitly identified to the contrary. I declare that I have used no other sources and aids other than those indicated. This work has not been submitted elsewhere in any other form for the fulfilment of any other degree or qualification.

Bonn, December 9, 2019 _____

Sara Hahner