11th International Conference on Customization and Personalization MCP 2024
The Power of Customization and Personalization in the Digital Age
September 25-26, 2024, Novi Sad, Serbia

AN APPROACH TO USE GENERIC DATA SETS FOR NEURAL NETWORKS IN PRODUCT DESIGNS THROUGH GEOMETRIC ABSTRACTION VIA PRIMITIVES

Manuel Ott[1], Niclas Meihöfener[1], Iryna Mozgova[1]
[1]Paderborn University, Chair of data management in mechanical engineering, Paderborn, Germany

Abstract: *Generative design suggestions and topology optimizations can help to reduce iterative process loops between calculation and design departments during product development processes. However, precise topology optimizations are computationally intensive, while generative designs benefit from swift suggestions to address design problems efficiently. Using artificial neural networks (ANN) can address this contrast of pre-defined aims by predicting topology-optimized designs, thereby combining both advantageous features. However, a challenge in Mass Customization is, that ANN are usually trained on specific geometries, making transfer to other applications impractical or requiring the creation of new datasets, which is economically unfeasible. Authors have already demonstrated a solution in other publications: dividing a geometry into geometric primitives like cuboids to perform abstraction. An ANN can then be trained to recognize optimized cuboids, which can be assembled back into a complete geometry, comparable to the finite element methods, which divide geometries of parts in finite elements enable mechanical property calculation. This publication aims to illustrate the steps of the approach in which the complete geometry of a part is segmented into these primitives, and the benefits obtained. Various methods will be explored, including automated workflows on modern low-code platforms, to enable generalized use.*

Key Words: ***Generative design; Topology optimization; Artificial neural networks; Product design; Product development***

1. INTRODUCTION

The desire for individuality and the need to distinguish oneself from others is inherent in human nature and therefore of essential importance. At the same time, given today's production volumes, it is a tremendous technical challenge to meet this desire for individuality within the constraints of quality, cost, and especially time, which necessitates a shift in technical thinking (Pfeifer, 2015). A high variety of options must be tailored to the needs of the consumer while still ensuring profitability. Simultaneously, sustainability issues are ever-present, requiring a more conscious use of existing resources in the form of efficient lightweight construction (Kupfer, 2022). It is precisely in this context that structural optimization comes into play, which is typically carried out in existing development processes by specialized, and thus limited and costly, personnel. In terms of mass customization this means, that for individual light weight products, structrual optimization needs to be much more efficient to meet the goal of effortable costs. Therefore, to ensure economic viability, these processes must be rethought and optimized (Georgiev, 2011; Zwettler, 2019).

In addition to sustainability, reports on artificial intelligence and its applications are another driver of modern visions, aimed at making human work more efficient or even replacing it altogether (Scheuer, 2023). Thus, it can therefore be inferred that the combination of utilizing artificial intelligence in structural optimization is indispensable for meeting the modern challenges in individualized product development.

2. PROBLEM DESCRIPTION

Ensuring the mentioned sustainability through efficient lightweight construction in times of increased individualization is only possible if design processes are automated or simplified to the extent that it becomes economically feasible to design a wide variety of products take into account the aforementioned factors. Paying attention to lightweight construction and possibly even carrying out initial calculations or structural optimizations in the early phases of product development, thus in the concept phase, appears unmanageable in many parts of companies (Zwettler, 2019).

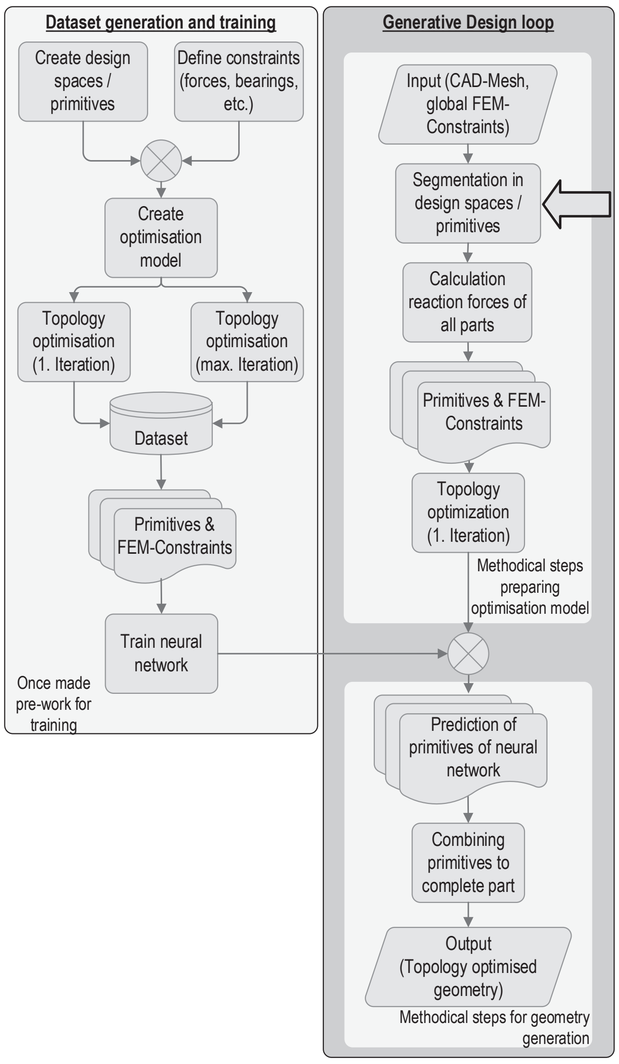
To automate such processes, artificial neural networks (ANN) can be employed, which, mostly require very data-intensive applications. Not least, this presents opportunities, which, based on various statistics, lead to improvements in product quality and process optimization. However, some surveys also reveal several hurdles. As shown in (Deloitte, 2023), in addition to the lack of competence and implementation challenges, data problems are a significant factor.

The latter issue of data problems is gaining increasing influence, particularly in fringe areas of applications, including in the context of structural optimization. An ANN requires a training dataset to perform tasks such as prediction, analysis, and clustering. This dataset is used to teach the ANN which input leads to which output. Large amounts of data are needed for this purpose, which, depending on the training method, must already be labeled or may be available in raw, unorganized form (Industrial-AI, 2021).

Various approaches to addressing these problems exist in the literature. On the one hand, ANNs can be used to generate numerical predictions for solving a mathematical matrix or to accelerate the corresponding algorithms. On the other hand, direct geometric elements can also be predicted. Depending on the objective, large datasets of various structurally optimized geometries are necessary to teach the ANN how the corresponding output should look under different assumptions (Woldseth, 2022).

3. PREWORK AND METHODICAL APPROACH

Based on the explained problem, it is of great importance to integrate ANNs and their possibilities to accelerate processes into structural optimization. This has the advantage that ANNs learn what structurally optimized geometries look like. Ideally, a network only needs to be trained once in order to be able to predict the subsequent optimizations. However, difficulties are to be expected when transferring to different geometries, as in (Ott et al., 2022) have already shown, which is why a method was developed that enables the generic use of a prepared data set through the geometric abstraction of a complex geometry by primitive geometries (Ott et al., 2023). This method is shown in figure 1. The method distinguishes between three areas: the one-off preliminary work to train the ANN, the methodical steps to automatically build the simplified optimization model and the subsequent merging of the previously optimized primitives into an overall geometry. By abstracting the overall geometry, the data set can initially consist of geometric primitives, which in the proof of concept are cuboids. These are stored in the dataset in various dimensions with various loads and their corresponding optimizations. The methodical steps for building the model first break down the overall geometry into the individual primitives. This step is marked with an arrow in the figure. More details about this building block and how it was implemented are described next. The reaction forces are then calculated for the primitives created and the individual optimization models are built up. However, only the first iteration needs to be calculated here; the fully optimized primitives are then predicted by an ANN trained on the dataset. In the final phase, the individual primitives are then linked by interfaces to form an overall geometry.



Dataset generation and training
Create design spaces / primitives
Define constraints (forces, bearings, etc.)
Create optimisation model
Topology optimisation (1. Iteration)
Topology optimisation (max. Iteration)
Dataset
Primitives & FEM-Constraints
Train neural network
Once made pre-work for training
Generative Design loop
Input (CAD-Mesh, global FEM-Constraints)
Segmentation in design spaces / primitives
Calculation reaction forces of all parts
Primitives & FEM-Constraints
Topology optimization (1. Iteration)
Methodical steps preparing optimisation model
Prediction of primitives of neural network
Combining primitives to complete part
Output (Topology optimised geometry)
Methodical steps for geometry generation

Fig. 1. *Prework methodical approach for AI-assisted optimisation*

4. SEGMENTATION TO PRIMITIVES

Segmenting the existing geometry into individual primitives abstracts a specific part into generalized equal segments This problem of generalization is not only significant in this application but can be found in nearly every industrial application. For example, parts and assemblies are often developed using a modular system (such as in the automotive industry) to use as many identical segments as possible and minimize the production and development of unique parts, as this would greatly increase the effort. Analogies can be drawn to the developed approach, where primitives are seen as components of this modular system, optimized and reassembled by an ANN. Since the method should be as automated as possible, ANN can be considered to directly handle the segmentation into primitives. Another way to achieve this is by defining a rule-based algorithm that examines existing geometry for potential primitives and creates them when the rule dictates. Therefore, these two approaches and the experiments and results conducted within this framework are explained to support decision-making.

4.1. Segmentation via neural networks

Selecting the ANNs for segmenting geometric primitives in the application case described in this study is limited by several constraints. The criteria include the CSG (Constructive Solid Geometry) basis and the outcome as a composite geometry made up of primitives.

The CSG approach, which is already successfully used for many years, describes a form of representation that essentially consists of generating a geometry using boolean operations on basic solids (spheres, cones, cylinders, rectangular solids, etc.). One of the greatest advantages here is that the entire history of how a geometry is created is stored in a logic process, which makes it possible to implement changes to the model very effectively. In today's CAD systems, however, this representation method is only rarely used, but the principle of the modelling method remains the same and is represented in the figure below (Agoston, 2005; Fraß, 2009; Watt, 2002).

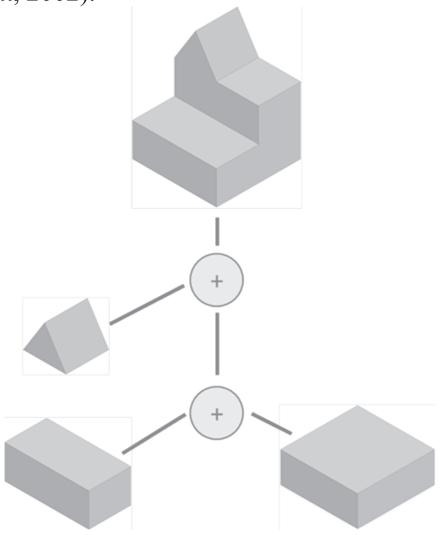


Fig. 2. *Representation of CSG concept*

Initially, an approach, which is based on former described CSG technique, published by (Ren et al., 2021) was examined. In summary, this approach attempts to convert a CAD input, represented as a point cloud, from a high-hierarchy CSG tree into a low-hierarchy CSG tree to ultimately remodel the geometry more effectively than traditional methods. However, this approach focuses on accurately and effectively remodeling the original geometry, while the presented work needs to focus on creation, subsequent access, and subdivision into primitives, which is not guaranteed in (Ren et al., 2021). Additionally, the presented in chapter 3 methodical approach is based on voxel model representation, which would implement an additional step of convertation from point clouds to voxels.

The wanted approach should be one, which only predicts the primitives that it finds in an existing geometry, without saving the corresponding CSG tree, so that only the primitives form the output and can be accessed. It is not expedient for this elaboration if the primitives are linked again at the end to form an overall structure without being able to access the intermediate steps, because this is precisely the point at which the generalisation of the geometry makes it possible to use a data set of the selected network for optimisation in a generic way.

(Yang et al., 2021) published a network architecture, which only tries to represent the given point cloud input in single surface mesh-based primitives. Point cloud segmentation does not matter much here, but simultaneously generated primitives do. The mesh learns the parameters of the primitives that it places over the point clouds without being monitored and can change these in order to represent the actual geometry more accurately. The scaling, rotation and transformation matrix are used for this purpose. In this approach, the primitives to be used must be specified and also an approximate number of them. The learning process was restricted for the investigations for the use case of this work to the extent that the rotation matrix should not be learnt, so that there are no rotations between the primitives. The procedure was tested on a chair model; in principle, the approach can be used to generate the primitives, but the predetermined number of primitives means that generic usability is only possible to a limited extent, but what causes a major obstacle in use is the lack of information on where and how the primitives touch, i.e., where transition areas are located. This is necessary at a later stage in order to define the forces and supports between the primitives. The occuring mistakes with this approach, which represent the reason for proceeding with a rule-based approach are shown in figure 3.

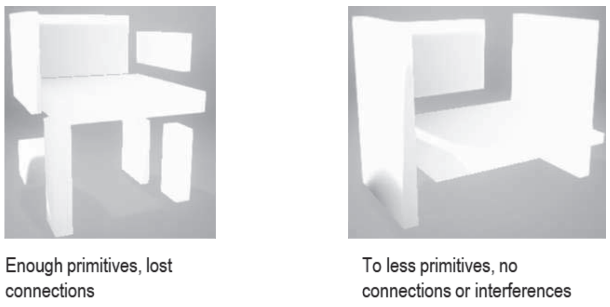
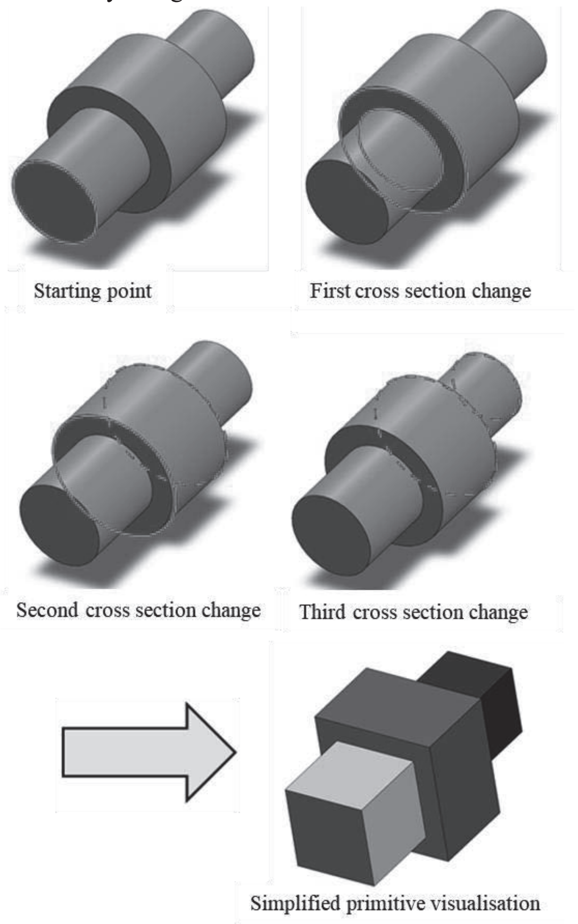

Enough primitives, lost connections
To less primitives, no connections or interferences

Fig. 3. Implementation of AI-based segmentation on example chair and occuring problems

4.2. Rule based segmentation approach

Generating the primitives according to certain rules is the most promising approach for the reason that all information can be saved during generation and used later. There is no black box behavior or lack of access to intermediate results. It is also possible for the user to change the parameters of the rules in order to make minimal improvements and specific adjustments. Cuboids were initially selected as primitives in this work. One reason for this is the time of use of the method defined in (Ott et al., 2023), which takes place very early in the development stage, where cuboids can be used to quickly define construction spaces and any geometries in a simplified manner and, on the other hand, the ANN operates on a voxel basis. Since voxels can only ever approximate a geometry, a kind of edge would result on non-planar surfaces, which could only be reduced by selecting an extremely high resolution. However, a high voxel resolution requires an extreme amount of memory, which causes problems when training the ANN. Since the output of this method also forms the basis for a reconstruction, whether manual or automated is not relevant here in the first instance, a compromise between resolution and memory requirements is therefore preferable. The primitives should also consist of extrudable base surfaces. Within this work, these are the aforementioned cuboids with rectangles as base surfaces, but additional shapes such as cylinders with circles as base surfaces or similar are also conceivable. The principle of segmentation developed for this application is based on the detection of a change in cross-section in one spatial direction. A loaded component is iterated through in voxel representation. If a change in cross-section is detected, a new primitive is created in this area based on the last change. These generated primitives can then be used for the subsequent steps of the individual model setups. The principle of segmentation is shown schematically in figure 4.



Starting point
First cross section change
Second cross section change
Third cross section change
Simplified primitive visualisation

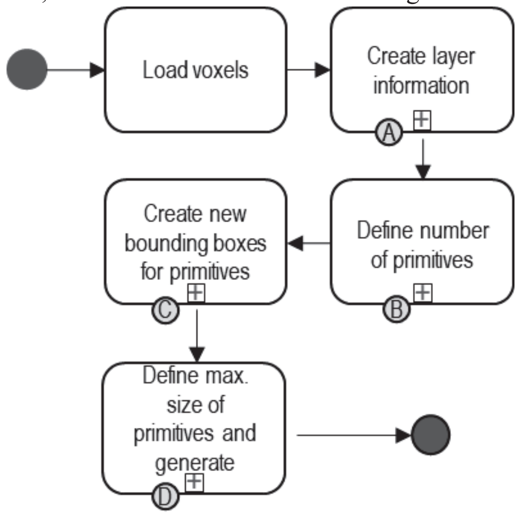
Fig. 4. The principle of segmentation

The additional major advantage over segmentation using ANN is the knowledge of the contact areas and positions of the primitives, as these are required for the subsequent conversion.

5. TECHNICAL IMPLEMENTATION OF METHODICAL WORKFLOW

All technical implementations of the method were carried out in the low-code platform *Synera.* Since some of the workflows here are very extensive and have a high level of detail, so it is difficult to visualize them in such an elaboration, the entire workflow is represented in the Business Process Modeling Notation (BPMN).

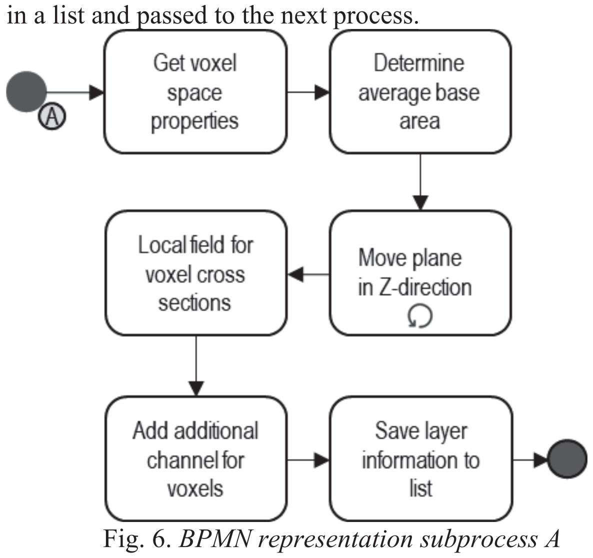
The developed method, as previously described, uses primitives to abstract optimization. Initially, the part geometry, already in voxel form, is loaded. Next, layer information of the geometry is generated to determine how many primitives need to be created. For these primitives, a new bounding box is generated, and then the primitives are finally created with the previously determined boundary dimensions. The overall main functions described and shown in figure 5 are marked with letters A, B, C and D, which represent the sub-processes, that are described in the following in detail.



Load voxels
Create layer information
A
Define number of primitives
B
Create new bounding boxes for primitives
C
Define max. size of primitives and generate
D

Fig. 5. *BPMN representation of main functions of rule based primitive segmentation*

Subprocess A is depicted in figure 6 and outlines the creation of layer information, which is essential for applying the principle described in chapter 3. Initially, the voxel space properties are extracted to identify active and inactive voxels. This data is used to determine the base area by averaging the number of voxels. Due to scaling, the Cartesian coordinates of edge lengths can differ with the geometry, so the voxel count on a plane is used to determine its cross-section. The plane is then shifted by one voxel row, initially described for the z-direction (figure 10), and the area averages are recalculated. To process the voxel space within the plane and their associated cross-sections in terms of changes, a local field is created to compute these changes. The relevant plane information and their changes are stored in a list and passed to the next process.



A
Get voxel space properties
Determine average base area
Move plane in Z-direction
Local field for voxel cross sections
Add additional channel for voxels
Save layer information to list

Fig. 6. *BPMN representation subprocess A*

The next subprocess B (figure 7), determines the number of primitives to be created. The threshold can be defined either by a relative change in voxels or an absolute value. The relative voxel change depends on the properties extracted in the previous process in the plane of consideration, while the absolute definition requires a fixed value. Both options can be used depending on the application and the geometry being analyzed to make the segmentation more precise in special cases. The voxel space, represented by the local field, is then filtered for changes using the preset threshold values. When changes are detected, the voxel space is split at those points in the plane, and the subdivisions are stored in a list. Finally, the length of the list is read. Since each change in the voxel space indicates a cross-sectional transition, a new primitive is created at each of these points, making the length of the list equal to the number of primitives to be generated.

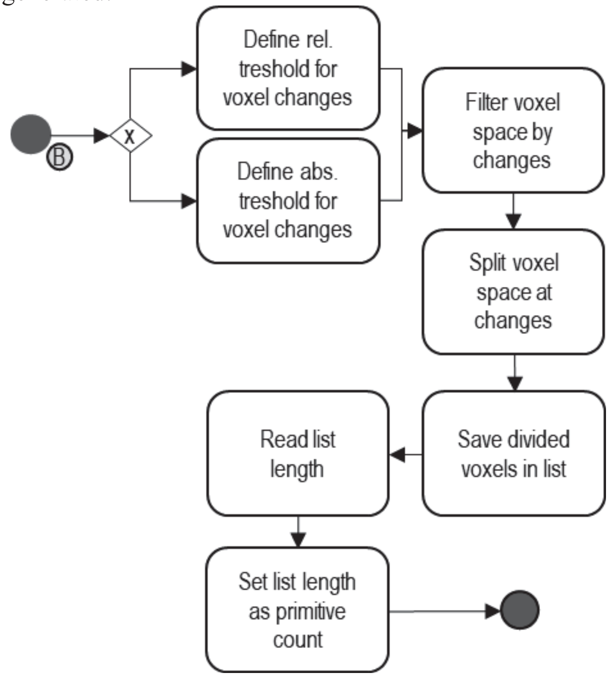

B
X
Define rel. treshold for voxel changes
Define abs. treshold for voxel changes
Filter voxel space by changes
Split voxel space at changes
Save divided voxels in list
Read list length
Set list length as primitive count

Fig. 7. *BPMN representation subprocess B*

In the third subprocess C (figure 8), new bounding boxes for the identified primitives are generated. Initially, the segmented voxel space is loaded, and a bounding box is defined starting from the base of the voxel space. The height of the bounding box is set at the location where the cross-sectional change was detected. A list of the new voxels within the corresponding bounding boxes is then stored. At this stage, the bounding box only has the base area of the voxel space and the height of the change as its dimensions, while the dimensions in the remaining two spatial directions are not yet defined.

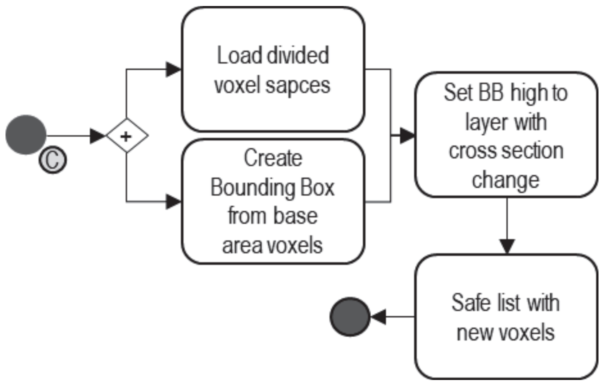

C
+
Load divided voxel sapces
Create Bounding Box from base area voxels
Set BB high to layer with cross section change
Safe list with new voxels

Fig. 8. *BPMN representation subprocess C*

The subprocess D is depicted in figure 9 and defines the maximum size of the primitives and generates them accordingly. First the voxel space properties are loaded, afterwards the upper and lower widths of the primitives are read from the former generated list. The maximum of both values is then defined as boundary boy width. This size is then extruded to the z-high where the cross-section change was detected. If the algorithm searches just in one direction, the new primitive is safed to the list then, otherwise bool operations of the different sections are done before.

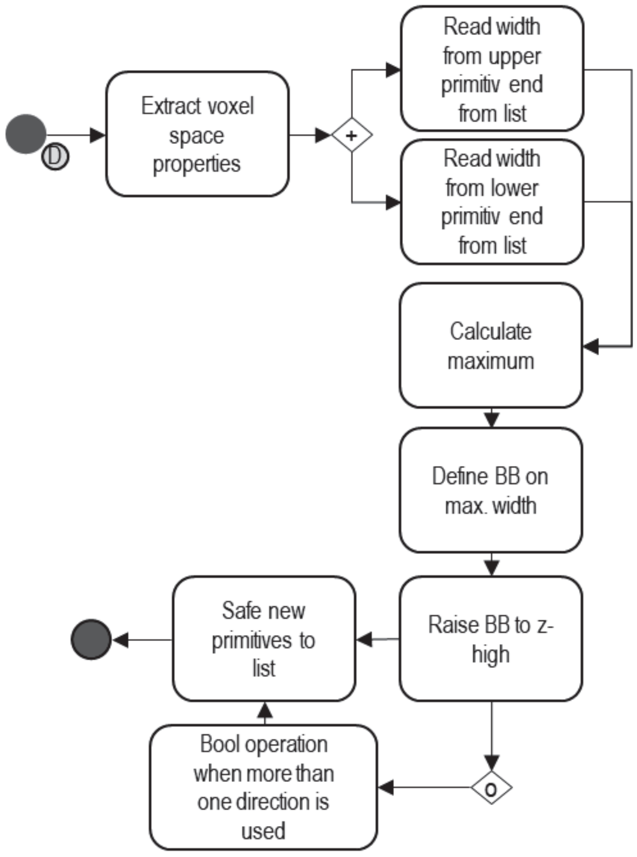

D
Extract voxel space properties
+
Read width from upper primitiv end from list
Read width from lower primitiv end from list
Calculate maximum
Define BB on max. width
Raise BB to z-high
Safe new primitives to list
O
Bool operation when more than one direction is used

Fig. 9. *BPMN representation subprocess D*

6. PROOF OF CONCEPT

In order to show, that the overall approach inclunding the segmentation algorithm works in generally, a proof of concept is presented in the following.

In this use case, the low-code platform *Synera* is used as a CAD environment, user interface and for part segmentation, software *Altair Optistruct* for topology optimization, Python package *PyTorch* for the CNN autoencoder, and AI platform *Weights & Biases* for monitoring the training process and hyperparameter tuning.

A mesh or voxelized object is used as the input for part segmentation. This input is converted into a centered and normalized voxel space with dimensions of 64x64x64 voxels. Subsequently, the gradient of the cross-sectional area is tracked for each voxel along the z-axis. Additionally, the number of discontiguous voxel spaces, defined as areas not connected to any other active voxels, is recorded. For each separate voxel space, the z-gradient serves as an indicator for new primitives. If the z-gradient exceeds a predefined threshold, a new primitive is created. Each primitive is represented by a cuboid encompassing all the original active voxels. Finally, each primitive is rescaled to fit into a 64x64x64 voxel space. The object maintains a maximum dimension of 64 units, with the other dimensions undergoing perspective-based rescaling to match the primitives of the training dataset. To ensure the connectivity of all optimized primitives, the intersecting areas between primitives are designated as non-design-space, i.e., active voxels. Figure 10 illustrates the workflow and results for an exemplary part, a fork head, segmented into primitives, including the network's input of the initial iteration of the individual primitives and their subsequent reconnection.

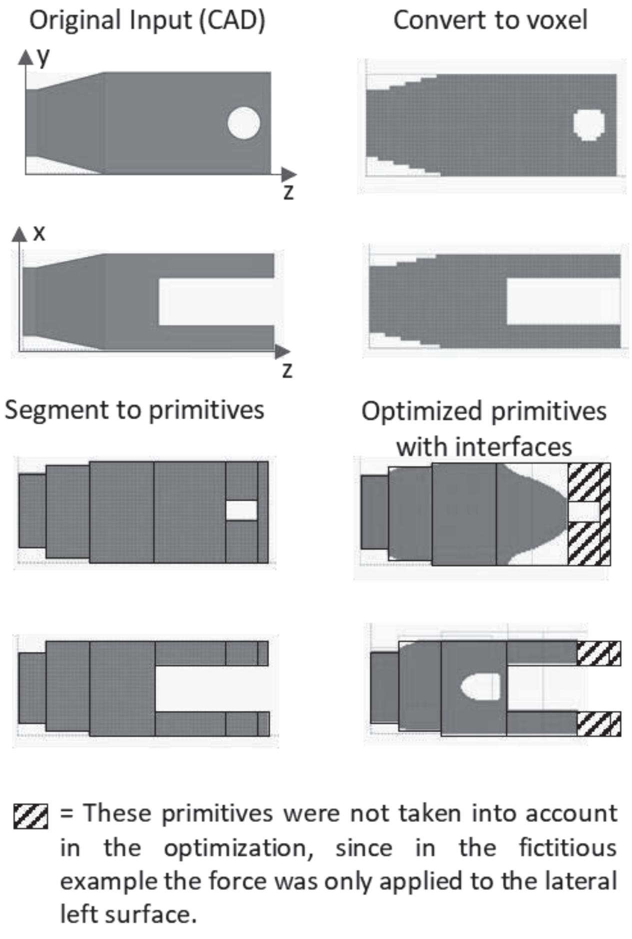

Original Input (CAD)
Convert to voxel
y
z
x
z
Segment to primitives
Optimized primitives with interfaces
▨ = These primitives were not taken into account in the optimization, since in the fictitious example the force was only applied to the lateral left surface.

Fig. 10. *Proof of concept on example geometry*

After the first iteration, the primitives are handed over to the ANN, which is represented by an autoencoder called U-Net. The Figure 11 shows at the top the given input (1st iteration of the primitives). The left bottom cornes represents the output of the ANN on voxel basis. Because it is hard to see the different depth in the voxel model, the model was transformed to a mesh surface, which can be seen at the right bottom corner. It is evident that the ANN can predict the individual primitives in an optimized form, employing structure-optimized patterns, as demonstrated by the visible supports of the upcoming interface in the right view.

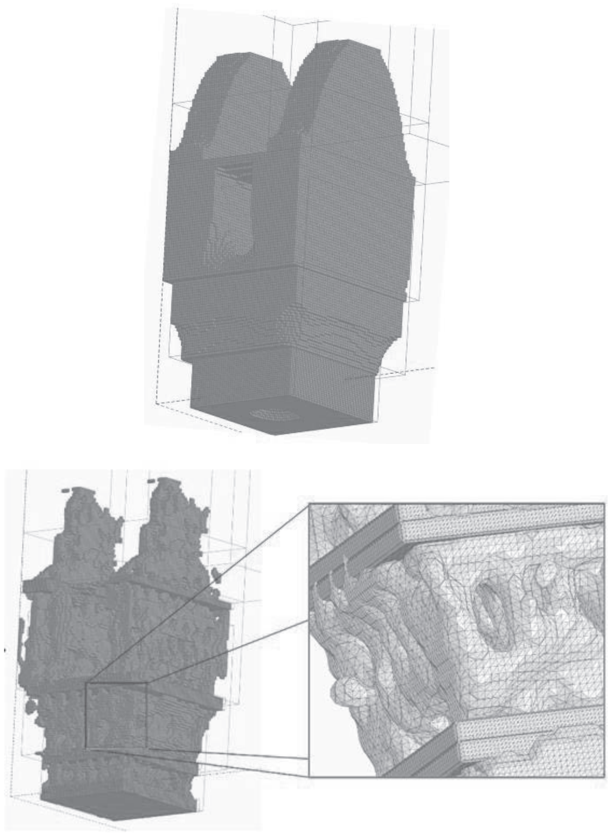


Fig. 11. *Prediction of neural network on example fork head*

In this example the overall time reduction compared between conventional topology optimisation and the developed workflow is about 77%.

7. CONCLUSION

Mass customization deals with the topics of customized product in high production volumes. Especially in time consuming development and design stages these highly individualized products, lead to high costs, which contradicts with the aim of cheap products due to mass production. In order to overcome this issue especially in terms of lightweight customized designs the structural optimization process needs to be revised.

Therefore, this study demonstrates how and why geometric models are abstracted and segmented into primitives, specifically in cuboid form, and why this is beneficial in the context of structural optimization using artificial neural networks to overcome the repeated generation of datasets. Initially, the topic is introduced, explaining how abstraction can be implemented using ANNs, which approaches can be used, and why they are unsuitable for this specific application. Subsequently, a rule-based automated workflow is presented, which scans geometries for cross-sectional changes and then generates new primitives based on the identified changes. Additionally, a detailed BPMN-modeled process is provided to clarify the exact procedure. Finally, the proof of concept is explained and illustrated with results from previous work to integrate it into the overall methodology.

In general, it can be argued that the most favorable approach depends on the specific application case. For the application case presented in the study, integrating ANNs into structural optimization, the rule-based approach is significantly better due to improved information transfer between individual primitives. While AI-based methods do provide efficient segmentation and CSG-based predictions, they lose the connections and orientations between these primitives which are essential for structural optimization. Future research can further explore how information and position transfer can be integrated into the structures of the tested ANNs. Additionally different aspects like the interface design for the connection between the primitives or the primitive's selection itself can be relevant research topics.

8. REFERENCES

Agoston, M. K. (2005): *Computer graphics and geometric modeling: Implementation and Algorithms.* Springer-Verlag London Limited, London, 2005 – ISBN 1-85233-818-0

Deloitte (2021): *Hemmnisse der Künstlichen Intelligenz in Deutschland im Jahr 2021.* Available from Statista from: https://de.statista.com/statistik/daten/studie/1297746/umfrage/hemmnisse-durch-kuenstliche-intelligenz-in-mittelstandsunternehmen/
[Accessed 26th January 2023]

Fraß, C. (2009): *Grundlagen der 3D-Modellierung*. Projektseminar, Technische Universität Dresden, 28. April 2009

Georgiev, C., Chakmakov, G., Todorov, G. & Nikolov, N. (2011): *Structural Optimization Methods in the modern product development process*. Wissenschaftliche Konferenz – Innovationen und Wettbewerbsfähigkeit, *Sofia, November, 2011.*

Industrial-AI (2021): *Stau auf der Datenautobahn.* News from 09.08.2021 from: https://ind-ai.net/industrielle-produktion/stau-auf-der-datenautobahn/
[Accessed 5th December 2023]

Kupfer, R., Schilling, L., Spitzer, S., Zichner, M., & Gude, M. (2022): *Neutral lightweight engineering: a holistic approach towards sustainability driven engineering.* In: Discover Sustainibility 3, Art. No. 17, Springer Link. Available from: doi: 10.1007/s43621-022-00084-9

Ott, M., Meihöfener, N., & Mozgova, I (2023): *Methodical Approach to Reducing Design Time by using Neural Networks in Early Stages of Concept Development.* In: 34th Annual International Solid Freeform Fabrication Symposium, Austin, Texas, 2023

Ott, M., Meihöfener, N., & Koch, R. (2022): *Neuronale Netze in der Konstruktion zur Ausschöpfung der Potentiale additiver Fertigungsverfahren.* In: 7. Tagung des DVM-Arbeitskreises „additiv gefertigte Bauteile und Strukturen", Tagungsband, Berlin, 8. + 9.November 2022. Available from: doi: 10.48447/ADD-2022-014
Pfeifer, T. & Schmitt, R. (2015): *Qualitätsmanagement – Strategien – Methoden – Techniken.* Carl Hanser Verlag, München, 2015 – ISBN 978-3-446-43432-5

Ren, D., Zheng, J., Cai, J., Li, J., Jiang, H., Cai, Z., Zhang, J., Pan, L., Zhang, M., Zhao, H. & Yi, S. (2021): *CSG-Stump: A Learning Friendly CSG-Like Representation for Interpretable Shape Parsing.* In: International Conference on Computer Vision 2021, Canada, Montreal, 2021. Available from: e-print arXiv doi: 10.48550/arXiv.2108.11305

Scheuer, S. (2023): *CEO Sataya Nadella sieht KI als zentralen Wachstumstreiber.* In Handelsblatt from 25.01.2023. Available from: https://www.handelsblatt.com/technik/it-internet/microsoft-ceo-sataya-nadella-sieht-ki-als-zentralen-wachstumstreiber/28942300.html
[Accessed 18th December 2023]

Watt, A. (2002): *3-D Computergrafik* – 3. Auflage. Pearson Studium, München, 2002 – ISBN 3-8273-7014-0

Woldseth, R., Aage, N., Baerentzen, J.A. & Sigmund, O. (2022): *On the use of artificial neural networks in topology optimisation.* In Structural and Multidisciplinary Optimization Vol. 65, Springer Vieweg. Available from: doi: 10.1007/s00158-022-03347-1

Yang, K. & Chen, X. (2021): *Unsivervised Learning for Cuboid Shape Abstraction via Joint Segmentation from Point Clouds.* In ACM Transactions on Gra-phics, Vol. 40 (4), Art.No. 152, August, 2021. Available from: doi: 10.1145/3450626.3459873

Zwettler, M. (2019): *Sechs zu vermeidende Probleme beim Einsatz von Simulation in der Konstruktion.* In Konstruktionspraxis from 27.05.2019. Available from: https://www.konstruktionspraxis.vogel.de/sechs-zu-vermeidende-probleme-beim-einsatz-von-simulation-in-der-konstruktion-a-833354/
[Accessed 25th January 2023]

SOFTWARE

Information on Synera low code platform are available from: **https://www.synera.io/**

CORRESPONDENCE

Manuel Ott, Research Assistant
Paderborn University
Faculty of mechanical engineering,
Warburger Straße 100
33098 Paderborn, Germany
manuel.ott@uni-paderborn.de

Niclas Meihöfener, Research Assistant
Paderborn University
Faculty of mechanical engineering,
Warburger Straße 100
33098 Paderborn, Germany
niclas.meihoefener@uni-paderborn.de

Dr. Iryna Mozgova, Prof.
Paderborn University
Faculty of mechanical engineering,
Warburger Straße 100
33098 Paderborn, Germany
iryna.mozgova@uni-paderborn.de