

AI-Based CT Data Pipeline

Robin Tenscher-Philipp¹, Tim Schanz², Martin Simon³

¹ Robin Tenscher-Philipp (M.Sc)
`robin.tenscher-philipp@h-ka.de`

² Tim Schanz (M.Sc)
`tim.schanz@h-ka.de`

³ Martin Simon (Prof. Dr.-Ing.)
`martin.simon@h-ka.de`

Hochschule Karlsruhe - Technik und Wirtschaft
University of Applied Sciences
Fakultät für Maschinenbau und Mechatronik
Moltkestr. 30
76133 Karlsruhe

Abstract. Data generation plays an increasingly important and crucial role in training artificial intelligence (AI) models. Focusing on the field of 3D computed tomography, for example, there is often a lack of available data. To address this problem, we propose an AI-based solution to generate training data by adding artificial defects to CT data. Our method enables the generation of large amounts of realistic defect-laden parts that can be used as training data for AI applications. By automating the process and adjusting the parameters, we can generate different defect types and distributions. To evaluate the generated results, this work trains a segmentation AI and applies it to unseen real-world data. This approach closes the gap in the availability of training data and enables the industry to use AI technology effectively.

Keywords: Data Generation, Sparse Data, Artificial Intelligence, Deep Learning, Industrial Computed Tomography, Defect Analysis, Autoencoder, 3D Segmentation

1 Introduction

In the realm of technological innovation, Artificial Intelligence (AI) and Deep Learning (DL) have emerged as transformative forces. However, their voracious appetite for data presents a formidable challenge that spans across diverse applications. Our focus centers on addressing this challenge within the domain of industrial computed tomography (CT) data analysis, where data scarcity has long hindered progress. This paper introduces an AI data-driven pipeline that leverages a single CT scan of an industrial additively manufactured metal component containing internal defects as its foundational data source. This pipeline comprises a sequence of steps, each crucial for reshaping the landscape of data-driven AI applications.

This research represents an important contribution in the domain of data provision for AI and DL applications, offering a novel solution to the persistent challenge of data scarcity. Our ability to generate diverse data from minimal inputs has the potential to improve significantly industrial CT data analysis and numerous other fields reliant on data-intensive AI models. One of the most challenging one is 3D-CT-Data

2 Related work

The current state of synthetic data generation research encompasses three major approaches: algorithmic data generation, simulation, and AI-based methods. It involves a diverse array of techniques and algorithms. Classical algorithmic data generation relies on randomness as its core principle. In this case, the data can be generated directly by algorithmic rules that are fed with random numbers to increase variation [1]. The ranges of random number generation must be parameterized for this purpose. For this, a deep understanding of the characteristics of the target data is necessary to set up and parameterize the corresponding rules. Depending on the complexity of the data to be generated, in this work industrial CT data, an algorithmic generation can become very complex, which can affect the realism as well as the variance of the data. Another way of generating data is simulation [2, 3]. Here, the real generation process is digitally recreated. Depending on the effort and the application, a simulation can provide real data close to reality. In combination with a controllable random generator, a lot of variant data can be generated. This generation can be used for industrial CT data. With the use of deep learning (DL) techniques, in particular autoencoders (AE) [4] and generative adversarial networks (GAN) [5], which can be used in many different applications. Generative networks are mainly used to generate data [6–11]. Compared to previous methods, DL has the advantage that artificial neural networks (ANN) can learn what the data should look like by using training data. The choice of training data is important. They should be sufficiently numerous and reflect all necessary real features as well as possible through their variance. In this way, good training conditions can be created in which the ANN learns the patterns and structures of the training data and can thus generate new data. Our approach of an AI-based generation of industrial CT data including defects is described below in this paper. The state of the art in data generation is an ever-evolving field of research. The field is constantly evolving and expanding, particularly with the advent of artificial intelligence methods and the data they require. The choice of a suitable data generation method depends on the application, the required know-how and the available hardware.

The landscape of segmentation in recent years has witnessed a significant transformation. Traditional methods, while still relevant in many applications, are progressively giving way to AI-driven approaches. Among these classic techniques is thresholding, which relies on variations in image intensity to isolate features within a specified intensity range. Its effectiveness hinges on features having distinct intensity values. When other features share the same intensity range, thresholding becomes incapable of distinguishing between them. For scenarios where features vary over time and require contextual information for recognition, conventional algorithmic techniques, such as edge detection, exhibit limitations due to their rigid rules. Deep Learning (DL) emerges as a game-changer in such situations. Through training ANNs acquire the ability to discern features within their natural context. When the circumstances change, the ANNs benefit from their ability to abstract and are still able to recognize the desired features. Important ANN architectures for segmentation are UNet [12, 13] based architectures. The basis of these architectures is their U-shaped structure based on encoders and decoders. The encoder part consists of convolutional layers that are used to extract the features. The decoder recombines the information in the ascending path and converts it into a representation. In addition to the AE, the UNet architecture adds skip links that connect the encoder and decoder in different layers. This links information in different layers to give a more accurate result. The UNet architecture is the basis for many other derivatives such as VNet [14] or PCUNet [15]. All these derivatives have different advantages depending on the data. Additionally, these architectures nowadays get enhanced using attention mechanisms [16] build in Transformer models like UNETR [17, 18].

3 Methodology

For an AI-based pipeline to generate data in the case of 3D industrial CT data of components with internal defects, the following pipeline Fig. 1 was developed.



Fig. 1. Pipeline from CT scan to image segmentation

(1) CT scan: The first step is to extract the internal defects from a real scanned component. The scanned part is available in voxel format. To extract the defects, the part is converted to STL format, which represents the surface geometry of the part. This step allows the extraction of the internal defects which are saved as individual separate STL files.

(2) Train Autoencoder: The second step involves creating the core of the defect generation process, an AE architecture as shown in Fig. 2. To work with the AE, the separate STL defects from the first step are transformed back into a constant-sized 25^3 voxel format. With this input data, the AE learns by encoding defects, mapping them to a compressed representation, and then decoding them to recreate defects in the voxel format.

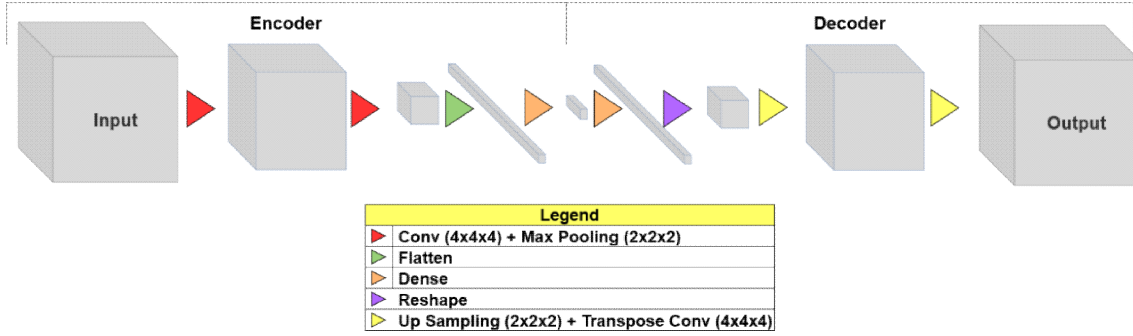


Fig. 2. Schematic of an AE architecture.

(3) Defect Generation: In the third stage, the trained AE is used to generate defects. This is done by separating the encoder part from the AE. By specifically varying the code layer, individually defined defects with desired geometric properties can be generated. To feed the correct activations into the code layer, individual properties are specifically analyzed, resulting in a parameter space that produces possible real generations. The larger the code layer, the more precisely individual properties can be mapped and set in the generation. Depending on the training quality of the auto-encoder, an infinite number of defects can be generated.

(4) Defect Placement: In the fourth step, the generated data is converted to STL format and placed into a desired STL object. The placement of defects in objects or components is done using algorithms based on properties such as number of defects, defect size and intersection. In this way, many variations of a component with different defect characteristics, numbers and distributions are generated and processed in the next step.

(5) CT-Simulation: In this work, the application and plausibility of the generation pipeline is evaluated using industrial CT data of components with inner defects. The CT simulation software ARTIST is used to generate the artificial CT data. Realistic simulation setups were created for this purpose. The software was automated using a script. The challenge lies in the parameterization of the software. Realistic setups require knowledge of X-ray physics. In a further step, the CT projections are reconstructed into a 3D image using the CT reconstruction

software Siemens Cera [19] By varying the simulation and reconstruction parameters, the diversity of the resulting images increases. As a result of step 5 of the step chain, artificially realistic CT images with internal defects are created. Additionally, ground truth is needed for the next phase, and to obtain this, the scans and reconstructions are idealized. This involves simulating a defect-free and defective component to generate binary ground truth through subtraction and thresholding. These label maps serve as training data for all variations of artificial CT data since defect positions remain consistent between ideal and realistic simulations.

(6) AI Segmentation: In the last step of the data generation pipeline, artificial neural network models, such as convolutional neural networks (CNNs), are trained with the synthetic 3D data. The 3D voxel data from (5) were sliced into smaller chunks with a resolution of 128^3 voxels to avoid hardware limitations. In this work a UNet architecture is used. Since this is a binary segmentation, the sigmoid function is used as the output function. The binary cross entropy is used as the loss. The performance of the segmentation is monitored using the metric Binary Intersection over Union (BIOU). After training, the model is used to segment real CT scans of components, e.g. for quality control in manufacturing. Manual annotation can then be omitted.


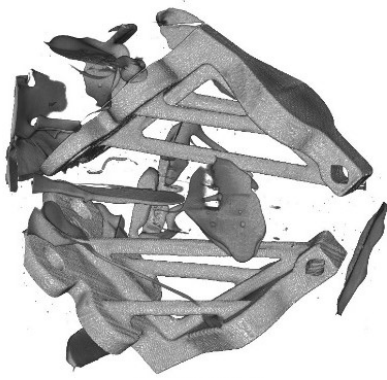
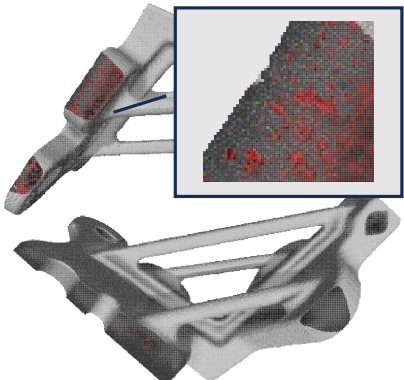
4 Results

In this paper we propose a pipeline that can synthetically generate single part features or whole objects. In our use case, we specialize the pipeline on defective industrial CT data. This pipeline can extract the information for the complete workflow from a single real image, train the AI and generate as much data as desired. The following are results of the pipeline and individual sub-points that are important and necessary to generate the desired synthetic data. The pipeline is developed as python modules for each step except for the simulation and reconstruction which makes it easy to automate the processes.

4.1 Initial CT Data, STL Component and Defect extraction

The first step in the Pipeline is to create a CT scan of the component (Table 1, left) as baseline. Only one scan is required for the whole pipeline which is done on a CT machine. The projection data is then reconstructed to a voxel 3D volume (Table 1, center). In this case two hock parts were scanned together using additional support material to fix the parts during the scan procedure. Marching cube [20] algorithm techniques allow to extract the material and defects surfaces depending on a threshold and a voxel resolution (Table 1, right). Additionally, the support material could be eliminated on STL level. By separating the hull as STL component it can be later used to place synthetic defect variations in it. The real defects itself are separated and stored one by one in an STL file as well. Those are the baseline for synthetic defect generation. Around 40,000 defects could be extracted from the component. By applying a filter to retrieve only watertight structures of a certain size around 25,000 defects are useful for further processing.


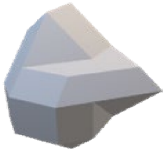








Table 1. (right) A photo of the 3D printed metal component. (center) Voxel rendering of 3D reconstruction after CT scan. (right) STL rendering of component with internal defects.

Real Component	3D CT Reconstruction	STL Component with Defects
		

4.2 Autoencoder Training and Defect Generation

Synthetic defect generation employs an AE. The initial step involves creating the AE architecture. To address the issue of varying vertex and face counts in STL defects, they are converted into binary voxel volumes at a consistent 25^3 resolution for easier data management (Table 2, 1st row). Approximately 25,000 voxelized defects serve as training data, with the encoder learning to create a simplified representation of the input in the code layer and then reconstruct it using the decoder. Post-training, the encoder and decoder can function as separate models. When the same data is applied to the encoder for prediction, it produces code layer representations. The code layer values for all 25,000 samples are recorded, and mean and standard deviation calculations are performed. These values are crucial for generating meaningful variations with the decoder. Values outside the mean and standard deviation ranges yield unrealistic results. The decoder can produce synthetic defects (Table 2, 2 d row) with higher resolution compared to the extracted defects, enhancing geometric details. This demonstrates the decoder's ability to generate geometries closely resembling real data. The rational number space offers an infinite number of variations, even when constraining values to the mean and standard deviation ranges for each code layer node. The binary voxel count of each generated defect is measured and stored, along with their corresponding STL versions, in a size-oriented database.

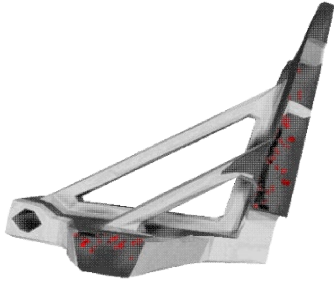

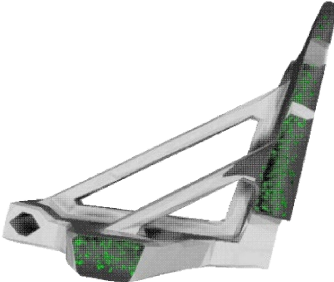
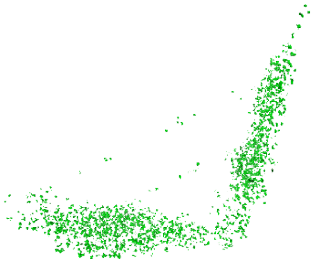
Table 2. (1st row) Extracted defects from the original component as single STL files. (2nd row) Synthetic defects generated with the encoder of the AE brought to application.

Extracted					
Generated					

4.3 Defect Placement Results

In 4.1 we preserved the STL surface hull of the component to place defect variations inside. To achieve this, an algorithm first loads the STL hull, which is the 3D model of the component. It then proceeds to calculate the specific defects to place within the component based on two key parameters: the desired average defect size and the standard deviation. These parameters guide the algorithm in selecting which folders within the defect database to load the defects from. In other words, the algorithm uses these parameters to determine the appropriate sources for defects that will be placed within the STL component. Also, depending on the inner volume in unit^3 and a defined defect density factor the algorithm calculates the number of defects to be placed. Afterwards it searches coordinates inside the STL hull by picking random coordinates, then checking if the coordinates are within the surface using the normal vector of the faces. If a coordinate is valid, it places a defect inside chosen randomly from the database within the defect size spread. The algorithm also allows to enable and disable intersections by using the empty hull or the defect updated hull for valid coordinate search. Additionally, multiprocessing could be activated to decrease the time for processing significantly depending on the available resources. Using our placement algorithm 40 variations of the component with defects are created. In Table 3, we compare the placement of defects. The first row shows large defects with a low density factor, while the second row shows small defects with a high density factor. With the first four steps it is possible to generate a large amount of STL Data with inner defects for further applications. Additionally, is it possible to place the generated defects in any STL.

Table 3. Comparison of defect placement using different parameters.

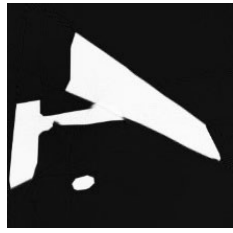
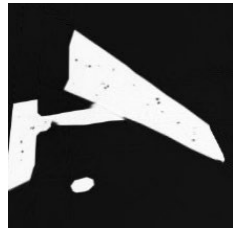
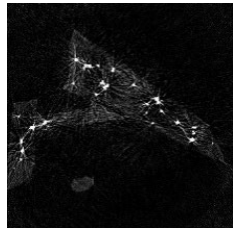
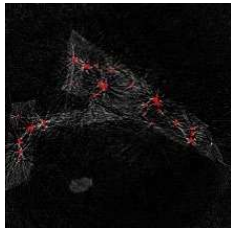

	Cross section of component with internal defects	Internal defects
Defects: 161 Density factor: 0.001 Avg. size: 5000vx		
Defects: 1700 Density factor: 0.01 Avg. size: 367vx		

4.4 CT Scan and Reconstruction Results

Creating useful training data for segmentation we need the samples and the corresponding ground truth. For this purpose, first a single ideal scan and reconstruction is done of the hull without defects and of each of the 40 variations of the defective components. By subtracting the reconstruction of the defective components from the hull only the defects and some random scan and reconstruction artifacts remain. Then a binarization is done to create the ground truth voxel

volume files. The process is illustrated in Table 4. Afterwards, the 40 component variations with defects are simulated and reconstructed more realistically using advanced parameters and functions of the simulation and reconstruction. Afterwards an algorithm is used to chop the volume samples and their ground truth into chunks, therefore it iterates through the sample. If the actual chunk has a certain amount of voxel above a material threshold and this amount is more than a certain percentage of the whole chunk, then this chunk is stored with the corresponding chunk of the ground truth. The selected chunk size is 128^3 . Using an overlap of 50% when windowing around 8000 samples for training could be retrieved.

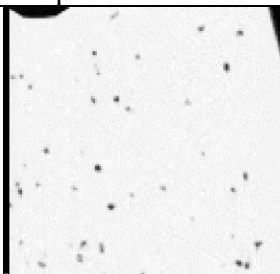
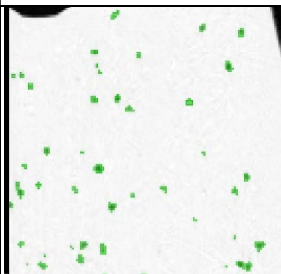
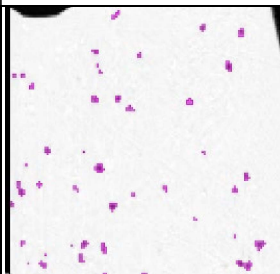
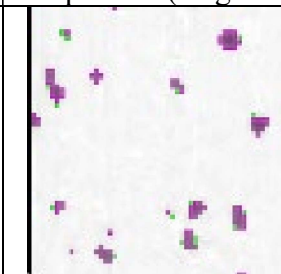
Table 3. Step by step illustration of ground truth preparation.

Hull without defects	Component with defects	Subtraction result	Thresholding	Binary segmentation
				

4.5 Segmentation Model and Training Results

The segmentation is done using a neural network model with the UNet architecture. Instead of building a model with full size CT volume resolution, the model is trained on the 8000 chunks with a resolution of 128^3 , split into train-, test- and validation data (80/10/10). This reduces the computational effort and training time. Since the model should focus on the features of the defects and their close surroundings of material to air transition the outer geometry is less relevant and therefore it is common to train on chunks. The model has around 1.5 million parameters. As layer activation the ELU function is used, as output activation sigmoid function. We use binary cross entropy as loss function. To track the progress of the training we use the metric BIoU. The model achieved remarkable 96% BIoU on the validation data. In the following table (Table 5) a sample slice of the synthetic test data is shown together with its corresponding ground truth, the prediction and a comparison between ground truth and prediction. We can see that the model was able to find all defects. The segmentation is almost complete only a very few voxel do not match. The comparison is magnified to increase the visibility of the differences.

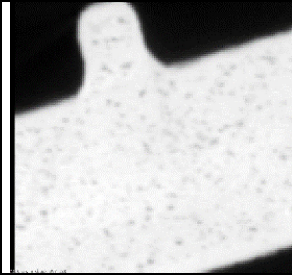
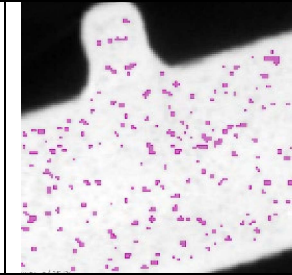
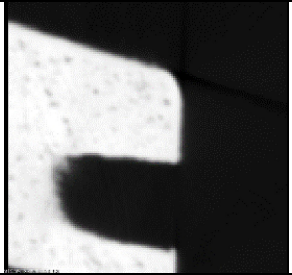
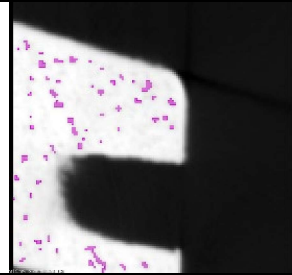
Table 4. Illustration of example sample slice, ground truth, prediction and a comparison overlay of ground truth and prediction

Sample	Ground truth	Prediction	Comparison (magnified)
			

4.6 Prediction on Real CT Data

Finally, the capabilities of the model trained on purely synthetic CT data and prediction on real data is shown. To predict on the real CT data, it was also chopped into chunks, fitting to the model architecture. In Table 6, we present the outcomes of our research, focusing on the final stage of our process. Our objective in this stage is to generate synthetic data with significant variability. This diverse dataset is intended for training a segmentation model used in quality analysis. Notably, we achieve this using only a single real CT scan as the basis for our synthetic data generation. We can see that the component has in some areas a very high porosity. The darker gray spots in the material are all defects. The trained defects segmentation model can detect most of the defects with good voxel accuracy in respect to the defect geometries. Thus, we show that the entire pipeline works. The synthetic data generated are sufficiently realistic to train segmentation AI models and apply them to real data.

Table 5. Prediction result on 2 real CT data sample slices.

Real sample 1	Prediction 1	Real sample 2	Prediction 2
			

5 Conclusion

The developed data generation pipeline provides a solution to fill any STL file with artificial defects (pores) that can be used as training data for various AI applications and also other applications which make use of STL data like finite element simulation. This pipeline provides new opportunities for the creation of numerous digital components with specific defect sets. In this work, the pipeline is used to generate data for defect segmentation in CT data. For conversion from STL to CT data we use the simulation tool aRTist and the reconstruction software CERA. By scripting and automating the pipelining steps, our method provides an efficient approach to generating CT data. The segmentation results with synthetic defects are used to evaluate the performance of our pipeline. The successful segmentation of defects in real data proves the effectiveness and stability of our method. This approach enables the generation of an infinite amount of data for various applications that require image data. This is especially valuable in industry, where data for quality assurance and other use cases is often sparse. Without sufficient training data, AI algorithms cannot learn effectively. Our pipeline addresses this problem by providing a wide range of defects with high variation, achieved by adjusting the code layer for pore generation. By combining simulation and reconstruction techniques, we also have control over the complexity and appearance of the defects. In summary, we have succeeded in developing a process that uses CT scans to generate countless realistic replicas from a single component and have demonstrated its quality in comparison with real data.

However, there are some steps in the simulation and reconstruction that require a lot of time and know-how for a good parameterization and thus for the generated results. To optimize, these tools need to be handled more efficiently. One of our next targets is to eliminate the need for external tools and to integrate the whole process into our AI-based pipeline. In addition, we are continuously working on extending the capabilities of our pipeline by adding new defect classes

such as cracks as well as creating a graphical user interface for this pipeline. This will greatly expand the range of applications and take them to a new level of versatility.

References

- [1] T. Schanz, R. Tenscher-Philipp, F. Marschall, and M. Simon, “Deep Learning Approach for Multi-Class Segmentation in Industrial CT-Data,” *ReJNDT*, vol. 1, no. 1, 2023, doi: 10.58286/28077.
- [2] Fuchs *et al.*, “Generating Meaningful Synthetic Ground Truth for Pore Detection in Cast Aluminum Parts,” *e-Journal of Nondestructive Testing (eJNDT)* 1435-49, vol. 9, 2019. [Online]. Available: https://www.ndt.net/article/ctc2019/papers/iCT2019_Full_paper_106.pdf
- [3] *aRTist - Analytical RT Inspection Simulation Tool*. [Online]. Available: <https://www.artist.bam.de/> (accessed: Jan. 17 2023)
- [4] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science (New York, N.Y.)*, vol. 313, no. 5786, pp. 504–507, 2007, doi: 10.1126/science.1127647.
- [5] I. J. Goodfellow *et al.*, “Generative Adversarial Networks,” Jun. 2014. [Online]. Available: <https://arxiv.org/pdf/1406.2661>
- [6] A. Sharma, O. Grau, and M. Fritz, “VConv-DAE: Deep Volumetric Shape Learning Without Object Labels,” Apr. 2016. [Online]. Available: <https://arxiv.org/pdf/1604.03755>
- [7] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling,” Oct. 2016. [Online]. Available: <https://arxiv.org/pdf/1610.07584>
- [8] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun, “A probabilistic model for component-based shape synthesis,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–11, 2012, doi: 10.1145/2185520.2185551
- [9] H. Huang, E. Kalogerakis, and B. Marlin, *Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces*: The Eurographics Association and John Wiley & Sons Ltd, 2015. [Online]. Available: <https://diglib.org/handle/10.1111/cgfl12694>
- [10] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction,” Apr. 2016. [Online]. Available: <https://arxiv.org/pdf/1604.00449>
- [11] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox, “Learning to Generate Chairs, Tables and Cars with Convolutional Networks,” Nov. 2014. [Online]. Available: <https://arxiv.org/pdf/1411.5928>
- [12] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox Olaf Ronneberger, *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46723-8_49#copyright-information (accessed: Dec. 6 2022).
- [13] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Image Processing, Computer Vision, Pattern Recognition, and Graphics*, vol. 9351, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III*, N. Navab, J. Hornegger, W. M. Wells, and A. Frangi,

- Eds., Cham: Springer International Publishing, 2015, pp. 234–241 ccessed: Apr. 27 202
- [14] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation,” pp. 565–571, doi: 10.1109/3DV.2016.79.
 - [15] T. Schanz, R. Tenscher-Philipp, F. Marschall, and M. Simon, “AI-Powered Defect Segmentation in Industrial CT Data,” *The Upper-Rhine Artificial Intelligence Symposium*, vol. 2022, pp. 2–11. [Online]. Available: <https://d-nb.info/1272864154/34#page=10>
 - [16] A. Vaswani *et al.*, “Attention Is All You Need,” 2017. [Online]. Available: <https://arxiv.org/pdf/1706.03762.pdf>
 - [17] A. Hatamizadeh and H. Roth, “UNETR: Transformers for 3D Medical Image Segmentation,” 2022.
 - [18] Ali Hatamizadeh, Dong Yang, Holger Roth, and Daguang Xu, *UNETR: Transformers for 3D Medical Image Segmentation*, 2021. [Online]. Available: https://www.researchgate.net/profile/ali-hatamizadeh/publication/350253170_unetr_transformers_for_3d_medical_image_segmentation
 - [19] *CERA – Software for High-quality CT Imaging*. [Online]. Available: <https://www.oem-products.siemens-healthineers.com/software-components> (accessed: Oct. 15 2023).
 - [20] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987, doi: 10.1145/37402.37422.