Deep Learning for Automating Additive Manufacturing Process Chains

Tobias Nickchen, M.Sc.

Faculty of Computer Science, Electrical Engineering and Mathematics
Paderborn University

This dissertation is submitted for the degree of
Dr. rer. nat.

March 2021

Danksagung

Ich möchte mich zu Beginn dieser Arbeit bei den Personen bedanken, die mich auf verschiedene Arten unterstützt haben und ohne die mir das Erstellen dieser Arbeit nicht möglich gewesen wäre.

Zu allererst möchte ich meinem Doktorvater Prof. Dr. Gregor Engels danken. Über den gesamten Zeitraum meiner Arbeit hat er mich immer unterstützt. Ich habe die regelmäßigen Besprechungen zu schätzen gelernt, die stets durch das Einbringen neuer Ideen und Sichtweisen und konstruktive Kritik geprägt waren. Vor allem in den letzten Monaten meiner Arbeit, hat mir die häufige Rücksprache bezüglich des Fortschritts der Dissertation sehr geholfen.

Ein weiterer großer Dank geht an die Phoenix Contact Stiftung, die es mir ermöglicht hat gemeinsam mit der Protiq GmbH an diesem industrienahen Thema zu arbeiten und dadurch neben der Arbeit an der Universität Paderborn auch tiefe Einblicke in industrielle Prozesse zu erhalten. In diesem Zuge möchte ich auch Dr. Johannes Lohn für seine engagierte und stets hilfsbereite Betreuung auf Seiten der Protiq GmbH danken.

Ein ebenso großer Dank geht an meine Familie, meine Freundin und meine Freunde. Meine Eltern haben mich mein Leben lang unterstützt und mir damit erst den Weg zu meiner Promotion ermöglicht. Meine Freundin Alina hat mich während meiner Arbeit an der Promotion immer unterstützt und mir den Rückhalt und die Ruhe gegeben, die ich zur Erstellung meiner Arbeit benötigt habe. Ein Dank geht auch an meine WG, die einen großen Teil meiner Promotion begleitet und mir die nötige Abwechslung neben der Arbeit verschafft hat.

Ich habe vor allem die angenehme und kollegiale Arbeitsatmosphäre mit meinen Kollegen des Direct Manufacturing Research Center (DMRC) und des Software Innovation Campus Paderborn (SICP) sehr genossen. Die Diskussionen mit den Kollegen aus den Ingenieurswissenschaften und der Informatik haben mir viele Einblicke und Ideen verschafft. Speziell möchte ich Dr. Stefan Heindorf für die gemeinsame Erstellung mehrerer Publikationen danken. Außerdem möchte ich mich beim Paderborn Center for Parallel Computing für die zur Verfügung gestellte Rechenarchitektur bedanken, ohne die die Durchführung meiner Experimente nicht möglich gewesen wäre.

Abstract

Additive manufacturing (AM) is a cutting-edge manufacturing technology that differs from traditional manufacturing technologies especially due to its high degree of flexibility. The production technique is characterized by its additive approach and its independence from forming tools. Therefore, an AM machine is able to produce arbitrary components with enormous geometric freedom. These benefits can be used by AM service providers to offer fast production of customized components for private and corporate customers. The flexible possibilities are accompanied by a very variable and daily changing process chain, which creates new challenges. In order to be able to fully exploit the benefits of AM even also with increasing production volumes, processes are needed that adapt automatically to the individual customer requests. For this purpose, data-driven solutions are needed. In the context of this thesis, we analyze the process chain of an AM service provider and identify potentials for further automation of the individual process steps and the intersections between the sub-processes. Only by automation of the daily varying processes an efficient production can be realized even with increasing production volume. We identified the two sub-processes of *Manufacturability Analysis* for AM and *3D Component Recognition* of additively manufactured parts as processes with high automation potentials. We show that data-driven solutions can be used to automate these processes that can not or only partially be automated using traditional algorithms. For each of the two sub-processes, we developed an individual solution based on 3D Deep Learning approaches that are able to adapt to the daily changing requirements using the existing process information. With our work, we have shown that data-driven process chains, such as the process chain of AM service providers, can highly benefit from data-driven solution concepts. Using our solutions, the level of automation of the process chain of AM service providers can be further increased, helping to ensure that the strengths of the AM technology can continue to be leveraged even as production volumes increase.

Zusammenfassung

Die Additive Fertigung (AM) ist ein modernes Fertigungsverfahren, welches sich vor allem durch seine Flexibilität von traditionellen Fertigungsverfahren unterscheidet. Durch den werkzeuglosen, additiven Ansatz ist eine AM Maschine in der Lage beliebige Bauteile mit enormen geometrischen Freiheiten zu produzieren. Diese Vorteile können von AM Dienstleistern genutzt werden, um die schnelle, individuelle Produktion von Bauteilen für Privat- und Firmenkunden anbieten zu können. Mit den flexiblen Möglichkeiten geht eine sehr flexible, sich täglich ändernde Prozesskette einher wodurch neue Herausforderungen entstehen. Um auch bei steigenden Produktionszahlen die Vorteile der Additiven Fertigung voll ausschöpfen zu können, sind Prozesse notwendig, die sich automatisiert an die individuellen Kundenanfragen anpassen. Für diesen Zweck werden datengetriebene Lösungskonzepte benötigt. Im Rahmen dieser Arbeit haben wir die Prozesskette eines AM Dienstleisters analysiert und Potentiale für eine weitere Automatisierung einzelner Prozessschritte bzw. Schnittstellen zwischen den Teilprozessen herausgearbeitet. Nur durch eine Automatisierung der täglich variierenden Prozesse kann eine effiziente Produktion auch bei steigendem Produktionsvolumen realisiert werden. Wir haben die beiden Teilprozesse der *Produzierbarkeitsanalyse* für AM und der *Bauteilerkennung* additiv gefertigter Bauteile als Prozesse mit hohem Automatisierungspotential identifiziert. In dieser Arbeit entwickeln wir datengetriebene Lösungen für diese beiden Prozesse, welche mittels traditioneller Algorithmen nicht oder nur teilweise automatisiert werden können. Für die beiden Teilprozesse haben wir jeweils eine individuelle Lösung auf Basis von 3D Deep Learning Ansätzen entwickelt, die in der Lage sind sich auf Basis der vorhandenen Prozessinformationen an die täglich ändernden Anforderungen anzupassen. Mit unserer Arbeit haben wir gezeigt, dass stark datengetriebene Prozessketten, wie die von AM Dienstleistern, von datengetriebenen Lösungskonzepten profitieren können. Durch den Einsatz unserer Lösungen kann der Automatisierungsgrad der Prozesskette von AM Dienstleistern weiter erhöht werden, wodurch dazu beigetragen wird, dass die Stärken der Fertigungstechnik auch bei steigendem Produktionsvolumen weiterhin genutzt werden können.

Table of contents

List of figures

List of tables

List of abbreviations

AM	Additive Manufacturing
API	Application-Programming-Interface
BJG	Binder Jetting
CAD	Computer Aided Design
CAM	Class Activation Mapping
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNC	Computerized Numerical Control
CNN	Convolutional Neural Network
CoM	Center of Mass
CT	Computer Tomography
CV	Computer Vision
DL	Deep Learning
DMRC	Direct Manufacturing Research Center
DNN	Deep Neural Network
EBM	Electron Beam Melting
FC	fully connected
FDM	Fused Deposition Modeling
FEM	Finite Element Method

FIFO	First in First out
FNN	Feedforward Neural Network
GAP	Global Average Pooling
GPU	Graphics Processing Unit
Grad-CAM	Gradient-weighted Class Activation Mapping
HCNN	Spatial hashing-based Convolutional Neural Network
HKS	Heat Kernel Signature
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
IR	Infrared
IT	Information Technology
LOM	Laminated Object Manufacturing
LMD	Laser Metal Deposition
LRN	Local Response Normalization
LRP	Layer-Wise Relevance Propagation
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MVCNN	Multi-view Convolutional Neural Network
NN	Neural Network
OBJ	Wavefront 3D object
OCNN	Octree-based Convolutional Neural Network
ORB	Oriented Fast and Rotated BRIEF
PBF	Powder Bed Fusion
PSH	Perfect Spatial Hashing

RBC	Random Brightness/Contrast Change
RGB	red, green and blue
RGB-D	Red, Green, Blue and Depth
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
RS-CNN	Relation-Shape Convolutional Neural Network
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
SM	Saliency Mask
SSE	Summed Squared Error
STL	Standard Triangulation Language
SLA	Stereolithography
SLS	Selective Laser Sintering
SLM	Selective Laser Melting
SVM	Support Vector Machine
TPS	Two-photon Stereolithography
UV	Ultra Violet
VDMA	German Engineering Federation
WACV	IEEE Winter Conference on Application of Computer Vision
XAI	Explainable Artificial Intelligence

Part I

Introduction

Chapter 1

Introduction

Additive Manufacturing (AM) is a relatively young and flexible manufacturing technique evolving between the traditional manufacturing methods. Its additive and layer-wise approach enables the production of complex geometries with extremely short time from ordering until delivery. AM service providers offer production services for private and industrial customers. The customer can upload his 3D model, which is subsequently analyzed, manufactured and send to the customer within a few days (see Fig. 1.1). The opportunities of this flexible and growing industry are accompanied by new challenges. Since the production process of AM service providers is extremely data-driven due to the processing of a wide variety of different customer orders every day, intelligent data-driven solutions for dealing with these challenges are necessary.

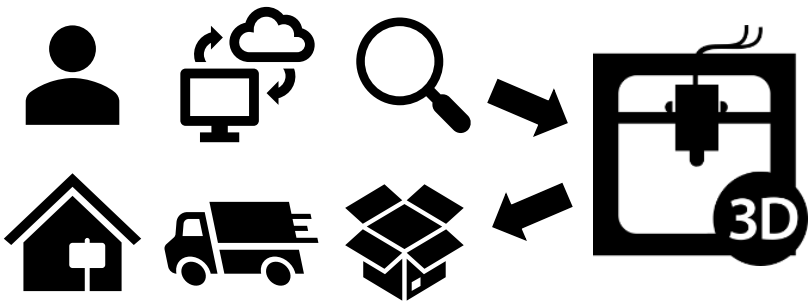


Fig. 1.1 AM process from customers 3D model upload until delivery.

This thesis is dealing with these challenges. The process chain of AM service providers is analyzed on the basis of the AM service provider Protiq GmbH. In our work, we are tailoring state-of-the-art research approaches from the domain of 3D Deep Learning (DL) with the specific requirements and restrictions of industrial applications. Automation potentials in the AM process chain are analyzed regarding the factors production time and costs, part

quality and failure reduction. Based on our studies, the possibility of using 3D DL algorithms for further automation is elaborated. Since the central element of the whole AM process chain is the digital 3D model, this information medium offers the potential of being utilized for the automation of different process steps throughout the process chain. The two sub-processes *Manufacturability Analysis* for AM and *3D Component Recognition* for additively manufactured parts which are highly related to the information of the digital 3D model, have been chosen as main focus of our work. Both processes require the analysis of 3D models and the recognition of decisive geometric properties that are relevant for the respective problem. Where traditional algorithms reach their limits, state-of-the-art 3D DL approaches are able to exploit the essential correlations through their data-driven behaviour. With our work, we show that complex industrial processes that are determined by combining the information of digital 3D models with physical processes can highly benefit from the use of flexible 3D DL algorithms.

The structure of this chapter is organized as follows: Section 1.1 gives an overview of the domain of AM and AM service providers. Based on that information, the motivation for our work is presented in Section 1.2. The approach of our research is explained in Section 1.3. In Section 1.4 the two main issues *Manufacturability Analysis* and *3D Component Recognition* are described. The solution concepts for dealing with that issues are explained in Section 1.5. Section 1.6 provides an overview of the disseminations which have been published in the context of our research and Section 1.7 describes the structure of this thesis.

1.1 Background

Additive or Direct Manufacturing describes the group of manufacturing techniques where components are manufactured using an additive approach. Components can be manufactured directly with the information of a Computer Aided Design (CAD) model without the necessity of e.g. a forming tool like it is necessary for injection molding. This saves time and costs. All AM techniques have in common that they are working in a layer-wise manner by discretization of the CAD models to layers and subsequently manufacturing a component by adding material layer by layer [33]. One of the most common techniques is the Powder Bed Fusion (PBF) process. The procedure is shown in Figure 1.2.

PBF machines can be used for the production of metal and polymer components. In both cases, the machines are working layer by layer. At first, a layer of material in form of powder is applied to the build platform by the powder leveling roller (See Fig. 1.2 left). Afterwards, a CO_2 laser is used to selectively melt or sinter the powder, which is preheated by an Infrared (IR) heater at the positions where the part should be generated. As the last

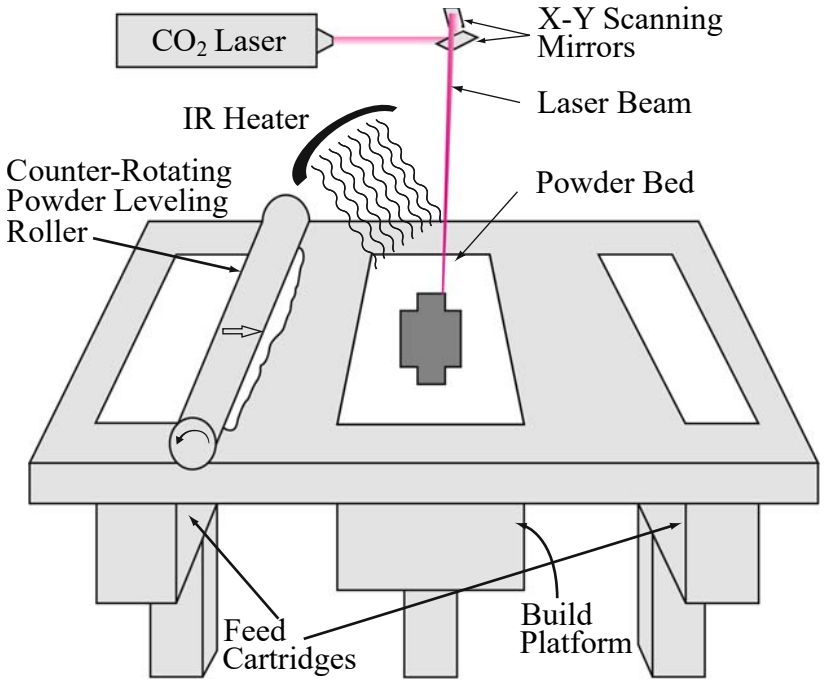


Fig. 1.2 Structure of a PBF machine [33].

step of the production of one layer, the build platform is lowered by one layer thickness and the process starts again. The cross-shaped geometry in dark gray in Figure 1.2 shows the current surface of the component that is produced.

Due to the additive approach, the procedure offers several advantages compared to traditional production methods like injection molding or milling, which are often used for series productions. Two of the main advantages are the high degree of geometric design freedom and the high flexibility due to the independence of forming tools. In contrast, for e.g. injection molding, a forming tool has to be produced prior to the actual production. This tool is used as a mould in which the material is moulded. The necessity of this forming tool leads to high fixed costs even before the production of the first end product. The high flexibility of AM enables the production of highly complex components optimized for their specific application which could not be produced without AM. The second characteristic allows AM machines to produce almost any part (within the dimensions of the machine) in the shortest possible time without the necessity for previous steps such as the costly and time-consuming construction of a forming tool. As a consequence, the price of an AM component is almost independent of the quantity to be produced.

These characteristics make AM an ideal solution for the provision of manufacturing services. Since only a digital 3D model is necessary for the production of a component, customers can easily transfer their 3D models to an AM service provider and order a

component. Without the necessity of tooling, the production time of arbitrary components is significantly reduced.

Due to its advantages, AM has been a rapidly growing industry with exponential growth of revenues in industrial production in the last decades. More and more companies in the manufacturing industry are integrating AM into their production processes and an increasing number of service providers offers services ranging from simple components to high-quality industrial parts. In the last ten years, the money spent on additively manufactured final parts has grown from 0.25 billions to around 1.5 billion Dollars [145]. Especially for AM service providers, this growth in production leads to new challenges in the entire AM process chain.

1.2 Motivation

With the growing interests and production volumes in the AM industry, also the challenges for AM service providers are constantly growing. One of the main criteria for the success of AM service providers is the flexible production, which makes it possible to reduce the time between the order of a 3D model by a customer and the delivery of the finished component to a minimum. This is possible as long as the individual processes in the process chain can be optimally executed. With the increasing production volumes, however, the internal structures of AM service providers are growing, too, leading to new challenges. The advantages of AM can only be exploited if the process infrastructure evolves as well. For such a flexible and dynamic production technique, this means that the associated process chain, including its individual processes, must be designed in a flexible and adaptive manner, too. This process chain can be divided into the physical process chain and the corresponding data process chain.

AM machine producers are constantly developing their machines in order to further strengthen the advantages described in Section 1.1. Also the machines for the necessary mechanical post-processing steps like depowdering or sand-blasting are automated to an increasing degree. Nevertheless, not only the machines itself but also the whole process chain of AM service providers has to be adapted to the growing needs. It is therefore necessary to consider not only the physical process chain but also the data process chain as well as the combination of both.

The high degree of individual diversity of customer orders requires flexible and fast processing of the associated data from the data upload to the final delivery of a component. Standard data processing solutions designed for series productions are not appropriable for that purpose.

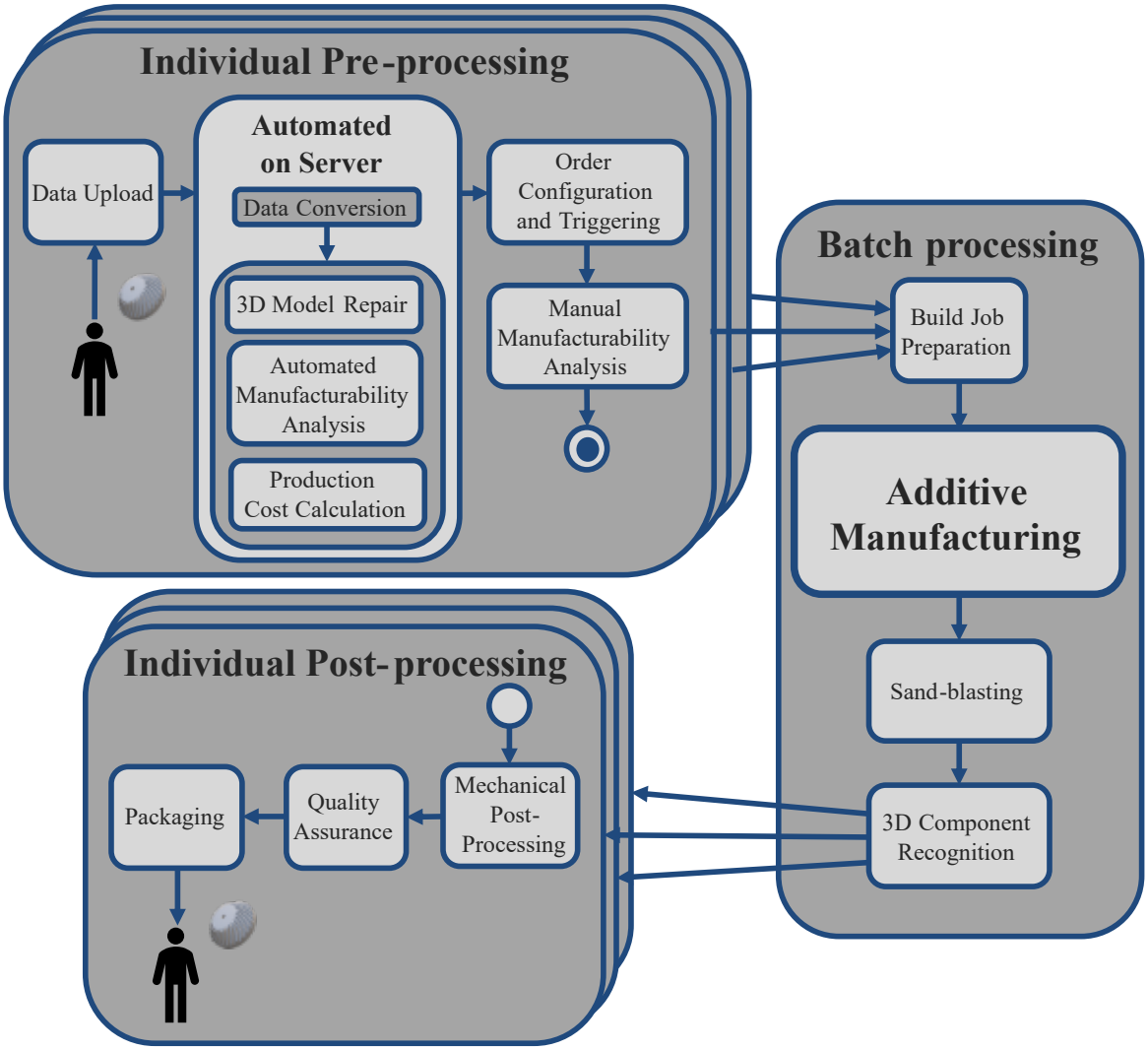


Fig. 1.3 Process steps in the AM process chain which are directly linked to 3D models [94].

For being able to analyze the internal processes at AM service providers, we first structured the process together with our reference AM service provider and summarized all the process steps that a component respectively the associated 3D model passes through in Figure 1.3. The properties of the digital 3D models influence nearly all process steps shown in Figure 1.3. After the upload of a 3D model by a customer, the 3D models are first converted from the CAD file type of the customers 3D model to Standard Triangulation Language (STL) files. The STL data type is the standard data type on which all processes related to AM are based [25]. Subsequent to the conversion, the models are repaired and the manufacturability is analyzed. Whether a model is manufacturable or not is defined by its geometry. In the last automated step, the production costs are calculated. The costs are directly related to the geometry of a 3D model, too. They are calculated for all possible

materials and post-processing steps. As soon as all necessary information are calculated, the customer can configure his 3D models and trigger the order. This step is followed by a manual analysis by the AM engineers of the AM service providers. An AM engineer is also performing the build job preparation. In this step, several 3D models are merged together into one build job which is then forwarded to an AM machine. For the 3D printing itself, the geometry of the 3D models is directly used to calculate the laser movement path in the AM machine. After the actual production, the components are cleaned from adherent powder in the sand-blasting step. The cleaned components subsequently have to be sorted again within the *3D Component recognition* step. This means that each component has to be assigned to its corresponding 3D model. After sorting, the components are post-processed for e.g. enhancing the surface quality. The last step before shipping is the quality assurance. In the quality assurance step it is again necessary to compare the exact geometry of the physical components with its digital counterpart.

Many of these process steps are still performed manually or semi-automated as it was not necessary to automate these process steps for small production quantities. Due to increasing production volumes, many of the current processes will no longer be feasible in the future. The entire process chain must therefore be optimized and automated in order to continue to exploit the benefits of AM. As explained above, the entire process chain is highly data-driven due to the daily varying customer orders and thus daily varying production. In order to be able to realize these extremely flexible and varying processes even with increasing production volumes, intelligent data-driven solutions for automation are necessary, which are able to adapt to the daily changing requirements. With our work, we are creating the connection of current state-of-the-art 3D DL approaches from the research domain with the requirements of a complex industrial production process chain.

1.3 Thesis Approach

The research presented in this thesis is based on the following approach.

1. At first an overview about the domain and the processes of AM service providers had to be created. Therefore, a process chain analysis was performed. The focus was on all processes that access the central element, the digital 3D model, and on the interconnection of these processes. On the basis of the analysis, the two sub-processes *Manufacturability Analysis* and *3D Component Recognition* were chosen as main topics for the further research.

2. A literature study on two topics was carried out: At first on the two sub-processes *Manufacturability Analysis* and *3D Component Recognition*. The current state of the art for the two sub-processes was considered from both the research and the industry perspective. Additionally, a literature study on 3D Machine Learning (ML) was carried out to develop an overview of possible innovative approaches for the two tasks.

3. For being able to design solutions for the two sub-processes, the requirements for possible solutions were defined from the industrial perspective of AM service providers and from the research perspective. The requirements from the industrial perspective differ from research requirements, since costs, computation time and usability play a much bigger role than in the research domain. Therefore, additional to the main requirements like the quality of the results, possible solutions need to fulfill these requirements for being applicable in industrial environments.

4. Based on the two previous aspects, appropriate approaches were designed. For the two sub-processes *Manufacturability Analysis* and *3D Component Recognition*, state-of-the-art 3D DL approaches were chosen which fulfill the requirements described above. Since the two tasks differ from classical benchmark tasks in the 3D DL domain, the approaches had to be adapted for our needs.

5. To prove the functionality, the approaches have to be tested using suitable data sets. Since there are no benchmark data sets for the two sub-processes *Manufacturability Analysis* and *3D Component Recognition* in the AM domain so far, synthetic data sets were generated. The generation of these data sets is of decisive importance for the reliability of the results. The data sets have to be aligned with real world data from AM service providers in order to achieve reliable evaluation results. With these synthetic data sets the capability of the chosen approaches to solve the tasks *Manufacturability Analysis* and *3D Component Recognition* were evaluated.

1.4 Problem Statement

As stated in Section 1.2, the different properties of AM compared to most traditional manufacturing techniques lead to additional challenges in the process chain of AM service providers. We have identified the two sub-processes *Manufacturability Analysis* and *3D Component Recognition* as promising processes for further automation using 3D DL methods. In this section, we will briefly explain the two steps.

1.4.1 Manufacturability Analysis

Due to the advantages of AM described in Section 1.1, AM service providers have to deal with many orders from different customers. These orders and the corresponding components are varying every day. Since many of these orders contain components which have not been produced before, it is necessary to verify that they are manufacturable using AM. However, in contrast to many traditional manufacturing processes, the definition of which geometries are manufacturable and which are not, is not trivial.

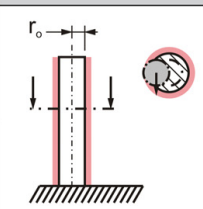
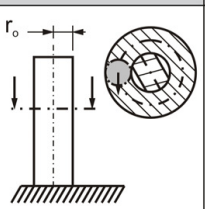
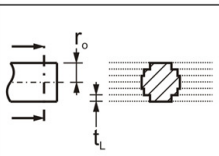
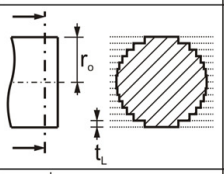
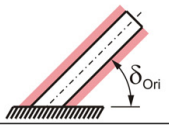
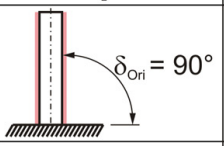
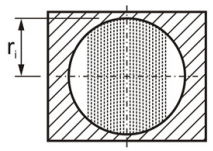
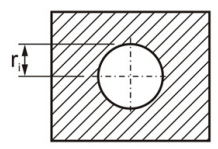
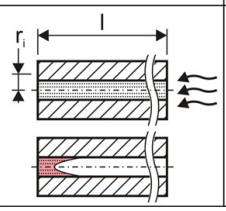
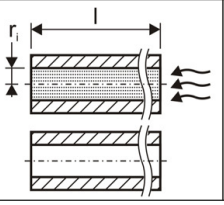
Group	Typ	Attribute	Description Regular Special	Design for manufacturing Unsuitable	 Suitable	LS	LM	FDM
Basic elements	Cylinders / Bores	Outer radius	Cylinders' outer radius should be large enough to structure each part layer with a boundary-line and enclosed raster-lines to minimize dimensional deviations and to avoid defects. LS: $r_o > 0{,}6$ mm LM: $r_o > 0{,}3$ mm FDM: $r_o > 1{,}5$ mm			X	X	X
			If their curvatures are mainly approximated by layers cylinders' outer radii should be as large as possible in order to decrease the approximization error related to the nominal outer radius.			X	X	X
		Orientation	Cylinders should be oriented orthogonally to the building plane to achieve the smallest possible dimensional deviations.			X	X	X
		Inner radius	Bores' insides that regularly require support structures can be manufactured without support structures if the bores' inner radii are small enough. LM: $r_i \leq 4{,}5$ mm FDM: $r_i \leq 5{,}0$ mm				X	X
		Length	Bores' lengths must be short enough to enable a sufficient removal of powder which is contained insed the bores. LS: $l \leq 8 * r_i$ LM: $l \leq 400 * r_i$ (Maximum tested ratio)			X	X	

Fig. 1.4 Example for design rules for the basic elements cylinders and bores [2].

An important aspect of the business model of AM service providers is that customers receive immediate feedback during the configuration and ordering process to enable a fast ordering process without delays. With regard to the *Manufacturability Analysis*, this means that a customer needs to get an immediate feedback if his component is manufacturable or

not. Therefore, an automated tool is necessary which directly generates a feedback after the upload.

For being able to evaluate if a component is manufacturable or not, a definition of manufacturability is needed. Due to the additive approach, generating that definition is a non-trivial task. With the growing interest in AM in the last years, several researchers worked on possible definitions. The most common approach is the usage of basic elements like cylinders, bores or cuboids for the generation of so called *design rules* based on these elements [1, 2, 62]. Some examples can be seen in Figure 1.4. The manufacturability of a cylinder is dependent on its radius, the manufacturability of a bore is dependent e.g. on its ratio of length to radius. More complex design rules also depend on the combination of different basic elements.

These design rules are a good guideline for an AM-compatible design of new components. Nevertheless, the design rules are not suitable for the verification of manufacturability for existing 3D models. The 3D models would have to be approximated by a combination of basic elements for making the design rules applicable. Since AM is often used for the production of complex geometries, it is not possible to generate a precise approximation of the corresponding 3D models. Therefore, the developed design rules are not applicable for an automated manufacturability analysis of existing 3D models.

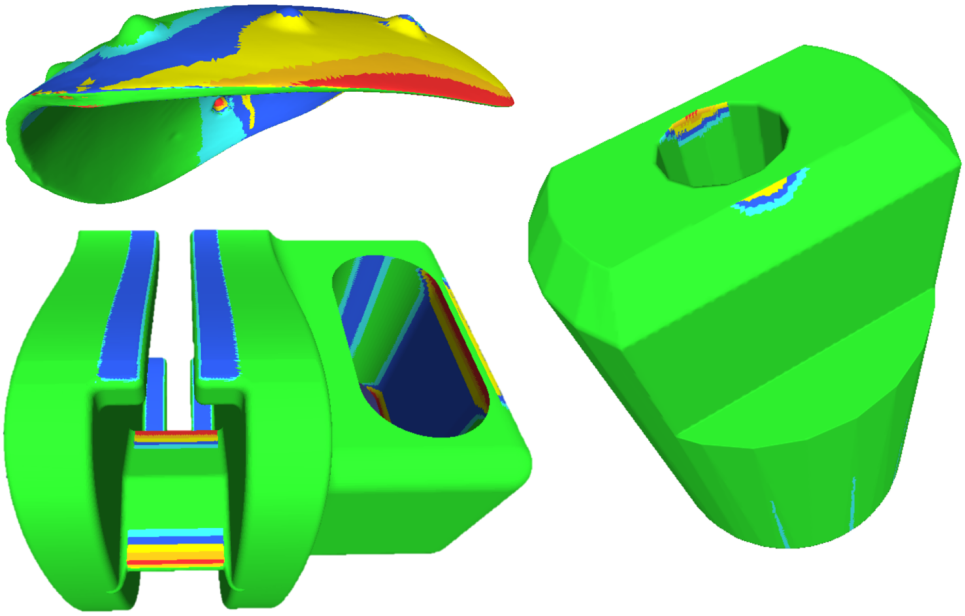


Fig. 1.5 Minimum wall hickness analysis shown for three example 3D models. Walls beneath a threshold of 0.3*mm* are marked in red. Areas slightly above the threshold are colored from orange over yellow to blue, non-problematic areas in green.

Currently, if at all, only tools for the validation of the *minimum wall thickness* criterion are used by AM service providers. This criterion is the most common problem in the AM domain. The quality of evaluations of current tools is often unsatisfactory which is why they are rarely used. Figure 1.5 shows some example 3D models with thin walls marked from blue over yellow to red. The tools often provide mis-classifications when the models include edges close to 90 degree. On the right side of Figure 1.5 an example of an erroneously marked thin wall can be seen.

Another method to validate the manufacturability of 3D models is the simulation of the build process [9, 125, 124]. Especially in the field of metal PBF, this method is used to simulate the build process and predict possible build failures. In contrast to design rules, simulation based methods are able to predict complex failures which occur e.g. due to the thermal behaviour of the used material. A drawback of these methods is the relatively high computation time which prevents their use for real-time evaluation as it is needed for our use case.

Due to the described disadvantages, neither of the two approaches is suitable for the automated real-time *Manufacturability Analysis*. Currently the manufacturability has to be validated manually by AM engineers (see Figure 1.3). Therefore, customers do not get an immediate feedback and the process from ordering to delivery of the components is delayed if a component needs to be modified. To solve this issue, an automated approach is needed that is capable of robustly evaluating the 3D models in real time.

1.4.2 3D Component Recognition

The necessity of *3D Component Recognition* arises from the batch production. Batch production implies that different components are manufactured simultaneously in the build chamber. This method can in principal be used by all different AM techniques. However, especially for Selective Laser Sintering (SLS) machines, the batch production is a particular characteristic. In contrast to other AM techniques the components which are produced, can be nested not only two-, but three-dimensionally in the build chamber (See Figure 1.6). This allows many different components to be built three-dimensionally nested in one build job. This characteristic leads to the challenge of *3D Component Recognition*.

At AM service providers, the orders of many different customers are processed every day. For optimal utilization of the SLS machines, components which shall be produced using the same material are merged together to one build job (See Figure 1.6). Therefore, most build jobs contain many different parts from different customers. During the build process the components are additively produced inside the powder. This procedure results in a so-called powder cake which contains the produced components without any fixed connection (See

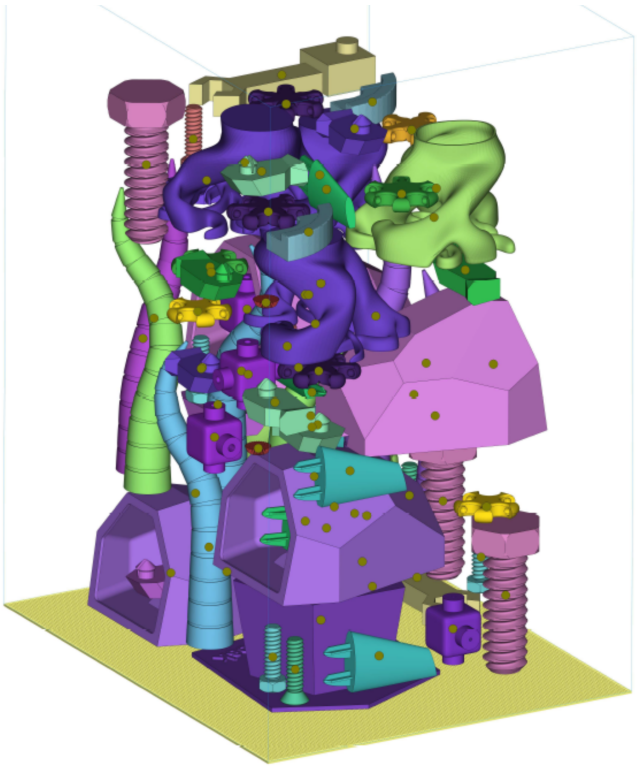


Fig. 1.6 An example build job for a SLS machine showing the batch processing character of SLS.

Figure 1.7 left). For further processing, the powder has to be removed from the components in the sand-blasting step (See Figure 1.7 right). After the sand-blasting, the components are passed on to the next post-processing steps as an unsorted group.

At this process step, the process chain switches from batch processing to individual processing again (see Figure 1.3). It is essential to know for which component which further processing steps has to be performed. The components must therefore be recognized and assigned to the correct orders again. For small production capacities that step can be performed manually by a visual comparison of the produced components and the corresponding set of digital 3D models. However, with increasing production volumes, this is not feasible anymore and an automated solution is necessary. A solution system must be able to automatically create the link between the digital build job data and the physical objects.

1.5 Solution Concepts

Our main contribution is the development of approaches for solving the two issues *Manufacturability Analysis* and *3D Component Recognition* which are described in Section 1.4. Both

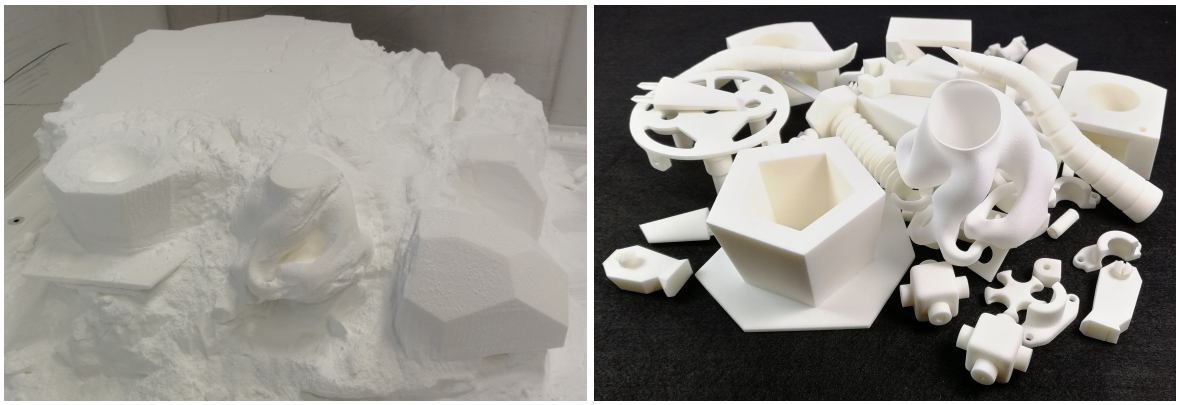


Fig. 1.7 The produced components in the powder cake after SLS production (left) and after sandblasting (right).

approaches should be able to solve the problems completely automated, without any need for manual intervention. Due to the high versatility in AM, especially at AM service providers, appropriately flexible and adaptable solutions are necessary. This versatility can best be managed using data-driven algorithms. Therefore, both solution concepts are chosen from the field of state-of-the-art 3D DL algorithms. Since both challenges, the *Manufacturability Analysis* and *3D Component Recognition* have different requirements, we have designed a suitable approach for the respective problem.

1.5.1 Manufacturability Analysis

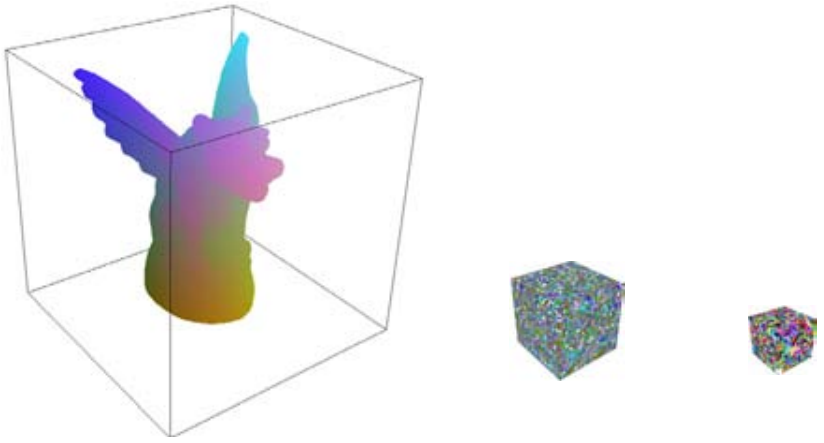
In section 1.4.1, we have explained that currently existing methods are not suitable for real-time evaluation of the manufacturability of 3D models for AM. On the one hand, design rules based on basic elements are not utilizable because the basic elements can not approximate the 3D models with sufficient accuracy. On the other hand, simulation based approaches provide a good analysis quality, but have the disadvantage of a high computation time what makes them unsuitable for the use case of real-time *Manufacturability Analysis* as well.

Instead of describing the complex interrelationships of geometric constraints by manually generated rules, we develop a data-driven approach that is able to use the daily generated production information. Through the data-driven approach, our solution is able to extract the information relevant for *Manufacturability Analysis* from the process data, especially that of production failures, and make it applicable for our application. Our developed evaluation system is to process the 3D models in high detail to be able to detect both small features as well as complex correlations which are relevant for the manufacturability. Additionally, the system is not only able to evaluate 3D models regarding their manufacturability, but also to generate feedback regarding critical areas in non-manufacturable 3D models.

Therefore, our system is split into two parts: The evaluation system and the feedback generation system. Based on a data set including manufacturable and non-manufacturable 3D models, the evaluation system is trained to detect critical features in the 3D models and predict if a 3D model is manufacturable or not. If a 3D model is rated as non-manufacturable, the feedback generation system calculates visual feedback in order to explain the decision making.

Evaluation System

As a basis for our solution, we have first chosen a suitable approach from the field of 3D DL on which we can build our solution. In the field of 3D DL, many different approaches for tasks like the recognition or segmentation of three-dimensional objects have been developed in recent years [146]. The main difference between the different approaches is the sensor type and the corresponding data they are using. Basically they can be divided into 2D and 3D data-based approaches. Since the whole geometry of a 3D model with its in- and outlying structures is relevant for the *Manufacturability Analysis*, image-based approaches can not be used for our purposes. They are not able to represent the inner structures of 3D models. Therefore, we have chosen to use the 3D data-based approach Spatial hashing-based Convolutional Neural Network (HCNN) [119].



Sparse 3D data
n=41,127 voxels in 128^3 volume

Hash table H
m=35^3 (=42,875)

Offset table Φ
r=19^3

Fig. 1.8 Example 3D data showing the effect of the efficient data storage of PSH [74].

The HCNN is a voxel-based 3D Convolutional Neural Network (CNN). Voxels are the three dimensional counterpart of 2D pixels. The main limitation of 3D CNNs is their high memory requirement and therefore the limitation of their resolution. The HCNN approach achieves the current state of the art with a resolution of 512^3 voxels while being used in batch

processing mode with a batch size of 32. The high resolution can be achieved using Perfect Spatial Hashing (PSH) [74] (see Figure 1.8), enabling the sparse 3D data to be stored and processed highly compressed.

Since not enough production data is available yet, the system is initially trained on the basis of data set which we synthetically generated. The data set contains 3D models which have been evaluated with regard to the criterion *minimum wall thickness*. The process steps necessary for calculation of the manufacturability rating based on a 3D model are shown in Figure 1.9. For calculation of the PSH representation at first a point-cloud representation has to be generated by a so called virtual scanner via ray-tracing [142]. The standard implementation of the virtual scanner generates a point-cloud of the surface of a 3D model. Since also the inner structures of 3D models are relevant for our analysis, we adapted the algorithm for being able to create point-clouds including the inner structures of 3D models. This step is followed by the generation of a voxel representation. The HCNN model is subsequently trained on the PSH data created from the voxels. After the training phase, the trained HCNN model can be used for the evaluation of further unknown 3D models.

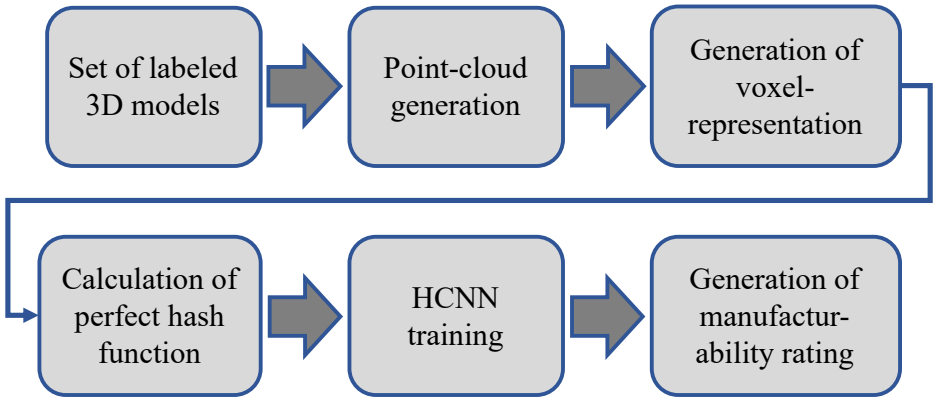


Fig. 1.9 HCNN data processing pipeline.

Feedback Generation System

Customers who upload a 3D model not only need a decision if their 3D model is manufacturable or not, but also information about critical geometries in their 3D model if a 3D model is rated as non-manufacturable. For being able to interpret the decision of the HCNN model, we added a feedback generation system (see Fig. 1.10).

We use the LRP [87] method to propagate features relevant for the decision back from the calculated output into the voxel-space. The LRP method has been developed for the creation of visual feedback for image analysis tasks. We have adapted the algorithm for application to

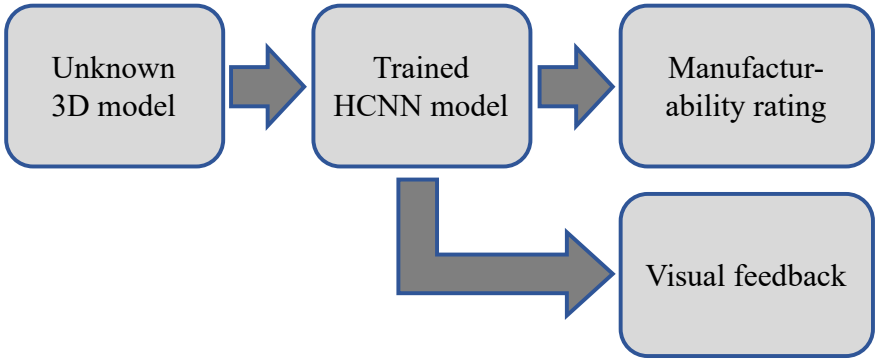


Fig. 1.10 HCNN manufacturability rating and feedback generation.

three-dimensional data. With that method a visual explanation of the most relevant features is generated which helps the customer to adjust the critical geometries of his 3D model.

Our approach provides a data-driven solution which is able to process 3D models with a high resolution and is able to learn relevant features from existing data. When the system is trained, it is able to process 3D models and generate a visual feedback regarding their manufacturability close to real-time.

1.5.2 3D Component Recognition

As explained in Section 1.4.2, a solution for the *3D Component Recognition* issue has to be capable to assign each physical component to the corresponding digital 3D model. Therefore, we searched for a solution, which is able to provide robust and accurate recognition of the produced components while being integrable into the process chain of an AM service provider. An important aspect here is that the system must be able to recognize completely different components on a daily basis. The system must be able to automatically adapt to the daily changes and generate all necessary information from the existing data of a build job and subsequently recognize the physical components using sensor data. All necessary steps must be directly integrated into the process chain and fully automated. Therefore, our work in the field of *3D Component Recognition* can be split into two parts: The recognition system itself and the data processing pipeline.

Recognition System

Since our research is based on the processes of AM service providers, we have conceptualized a possible physical structure of a sorting station as a basis for our general conditions. We envision the following structure of a recognition system. The system consists of a separation station and a conveyor belt with an integrated scanning area. After separation, the

manufactured components are transported by the conveyor belt through the scanning area where multiple sensors sense the component from elevated view points (see Fig. 1.11) and a recognition system assigns the physical components to the corresponding digital 3D models. The main challenge is that the system must be able to correctly recognize the components regardless of their orientation.

As mentioned in section 1.5.1, the field of 3D DL algorithms can be split into 2D and 3D data-based approaches. In the context of our work, we have evaluated approaches from both groups, where based on factors such as accuracy, computational costs and hardware costs, 2D data-based approaches have proven to be the most promising.

The central element of our solution is the multi-view-based approach RotationNet [56]. RotationNet is the current state of the art on recognition of the ModelNet40 classification challenge by Princeton ModelNet [146] which is used as benchmark for 3D object recognition approaches.

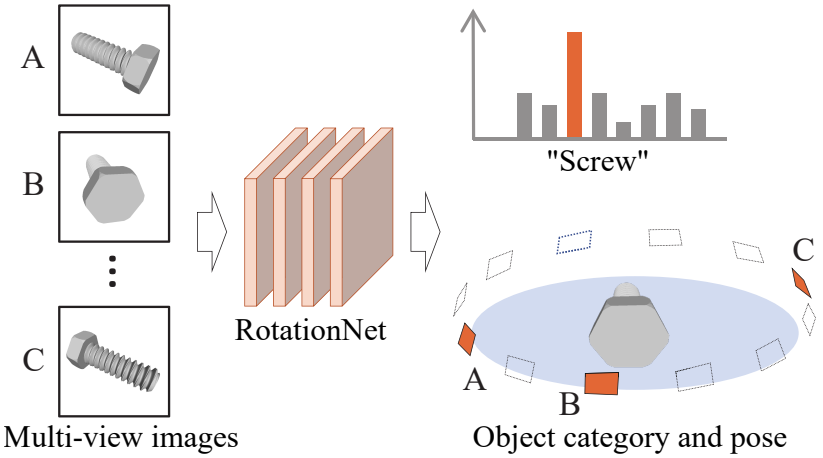


Fig. 1.11 Basic multi-view structure of RotationNet [56].

3D Component Recognition for AM differs from typical object recognition task by one crucial characteristic. Since the components are often unique, they only exist as digital 3D models prior to the actual production. Therefore, in contrast to standard tasks, no physical images are available which can be used as training data for the recognition system. In order to train a classifier, synthetic training data must be generated in the form of virtual images. When using synthetic training data, the accuracy of the recognition systems strongly depends on the adaption of the artificial data to the physical conditions in the inference phase. Therefore, an exact design of the training data generation method is significant [48].

The basic structure of RotationNet is shown in 1.11. In our envisioned physical setting, the system is using 12 cameras positioned in 30 degree steps around the z-axis with an elevation angle of 45 degree from the horizontal plane. The synthetic training data has to

be created with the same conditions. We render photorealistic virtual images using Phong shading [127, 73] with the same camera setting as used in the physical system.

Besides the photorealistic rendering itself, the components orientation in the rendering scene has a decisive influence on the accuracy of the recognition system. In the physical world, the components orientation in front of the cameras is determined by the laws of physics. During transportation on the conveyor belt, the orientation of a component is defined by the geometric form of the component and the gravity. Only a finite number of different orientations is physically possible after transportation to the scanning area (neglecting rotations around the z-axis). We designed an approach which makes use of the physical behaviour and calculates physically sound orientations based on the geometry of a 3D model. Using that approach, we are able to create synthetic training data which is optimally aligned to the physical situation

With the synthetic training data we generated in the explained way, we can bypass the need for physical photos and are able to train RotationNet for the recognition of physical additively manuactured components only using synthetic data.

Processing Pipeline

A solution for *3D Component Recognition* has to be directly integrable into the process chain of AM service providers. The procedure includes the steps shown in Figure 1.12. The necessary information for training the system must be generated automatically from the 3D models which are the central information source in the whole AM process chain. For each 3D model, first the physically sound orientations are calculated and subsequently used for the photorealistic rendering. As soon as the training data is available, the training itself can be started so that a trained instance of RotationNet is available when the production of the components is complete.

The information which components will be produced are available as soon as the build job preparation step is finished (see Fig. 1.3). In that step, a fixed set of 3D models is assigned to each build job. Quitting the build job preparation step can automatically trigger the calculation of physically sound orientations using the 3D models of the build jobs which can automatically be accessed from the process chain information. The time the AM process itself needs for the production of the components is used for the steps described in Figure 1.12. When the physical components reach the process step *3D Component Recognition*, a trained RotationNet instance is available for each build job.

With our solution we contribute to the automation of the *3D Component Recognition* step. The solution closes the gap between the research domain and industrial application.

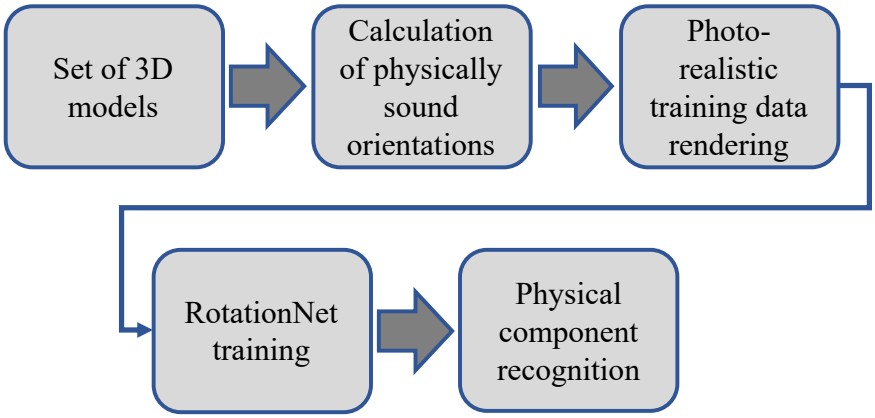


Fig. 1.12 The necessary steps for training RotationNet for the recognition of physical components based on a set of digital 3D models.

It is designed for a direct integration into the process chain of AM service providers and enables further automation of the process chain.

1.6 Overview of Disseminations

As part of our work, we have written several publications that describe the steps necessary to develop our solutions. Besides three publications for scientific conferences ([94, 95] and one in the revision phase at [132]), technical papers were published in industrial journals and presentations were given at industrial application conferences. In addition, current challenges on the way to a fully automated AM process chain were regularly discussed with the industrial AM service provider Protiq GmbH [101], within the working group *Process Chain Automation for AM* of the German Engineering Federation (VDMA) and with the members of the Direct Manufacturing Research Center (DMRC). The VDMA is the largest European association of companies and research groups in the field of engineering [141]. The DMRC is an interdisciplinary institute of the mechanical engineering faculty of the Paderborn University doing research regarding a wide range of topics related to AM and consists of twelve different chairs and 26 industrial partners [27].

Similar to the structure of this thesis, we first dealt with the analysis of the process chain of AM service providers. In [94], we developed a process model including the process steps which components respectively the corresponding 3D model are passing through in the process chain of AM service providers. Based on that model, we have been able to analyze which process steps offer the biggest potential for being automated using data-driven approaches. Using our model-based analysis and based on discussions with the members of the DMRC, we figured out that the two sub-processes *Manufacturability Analysis* for AM

and *3D Component Recognition* for additively manufactured parts are the most promising sub-processes for our further work.

In the following, we dealt with these two sub-processes. In a publication which is currently in the revision phase at the Rapid Tech conference [132], we dealt with the *Manufacturability Analysis* for AM. Our analysis of related work has shown that existing non-data-driven solutions are not or at most partially capable of providing an automated real-time *Manufacturability Analysis*. In our publication, we describe our own solution which is able to analyze 3D models regarding their manufacturability using the data-driven HCNN [119] approach and additionally able to provide a high resolution visual feedback regarding critical geometries using the LRP [67] approach. Our work regarding the *Manufacturability Analysis* has also been published in the engineering magazine *Konstruktionspraxis* [61] and presented and discussed at the technology days 2020 of the Phoenix Contact company [36].

Subsequently, we dealt with the sub-process *3D Component Recognition* for additively manufactured parts. The development of a first prototype solution, which provides important insights for the development of our final solution, was first published in two industrial journals [51, 53]. Based on the results of the prototype, a one-year project group of computer science master students carried out research on possible optimized solutions. The final solution and the corresponding results have been presented and discussed at the industrial Manufacturing Analytics conference [8]. Subsequently, our approach based on RotationNet [56] and optimized using physically sound training data has been published in the proceedings of the IEEE Winter Conference on Application of Computer Vision (WACV) [95]. Our approach provides a solution which achieves high recognition rates for the recognition of additively manufactured components using multiple cameras and is able to automatically adapt to the daily changing production.

1.7 Structure of this thesis

The structure of this thesis is shown in Figure 1.13. The thesis is split into the five parts *Introduction*, *Additive Manufacturing*, *Foundations and Related Work*, *Solution* and *Realization, Evaluation and Conclusion*. Subsequent to the introduction, in Part II, we first introduce foundations regarding AM and the corresponding process chain of AM service providers in Chapter 2. Based on these foundations, in Chapter 3, we describe the two sub-processes *Manufacturability Analysis* and *3D Component Recognition* which were chosen as reference processes in detail. For both tasks, we define the solution requirements which have to be fulfilled for the development of a suitable solution.

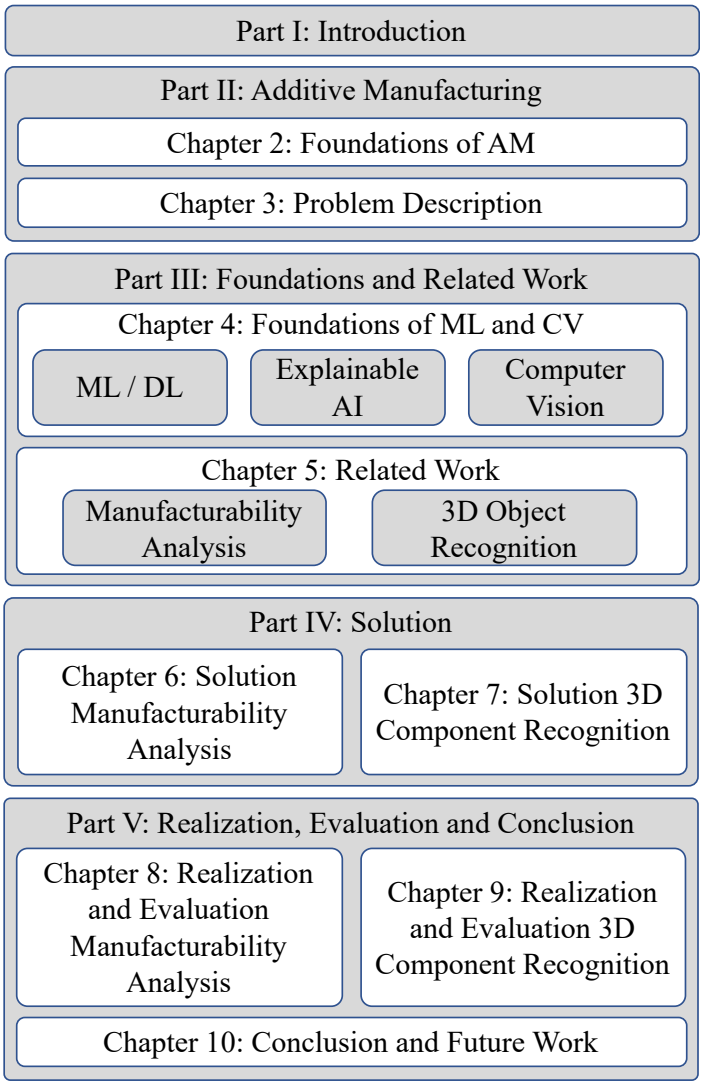


Fig. 1.13 Overview of the structure of this thesis.

Part III is dealing with foundations and related work regarding the two process steps *Manufacturability Analysis* and *3D Component Recognition*. In Chapter 4, we describe the foundations related to ML, DL and Computer Vision (CV). Additional to foundations regarding ML and CV itself, the field of Explainable Artificial Intelligence (XAI) is introduced.

In Chapter 5, we give an overview about the related work regarding the *Manufacturability Analysis* and *3D Component Recognition*. Section 5.2 deals not only with *3D Component Recognition* for AM but with the higher-level field of *Object Recognition*. In Section 5.1, we describe the current state of the art in the field of *Manufacturability Analysis* for AM and give an overview of *Manufacturability Analysis* in related domains.

The fourth part is dealing with the solutions which we developed. In the Chapters 6 and 7, we present the development of the solutions for the two main issues *Manufacturability Analysis* and *3D Component Recognition*.

In Part V, the realization and evaluation of the solutions for *Manufacturability Analysis* and *3D Component Recognition* is described in the Chapters 8 and 9. We first explain the realization of the solutions and the experimental setups. Afterwards, the experiments and corresponding results are presented and a discussion and summary are outlined.

In Chapter 10, we present a conclusion of the work in this thesis. We outline our main contribution and give an overview about future work.

Part II

Additive Manufacturing

Chapter 2

Foundations of Additive Manufacturing

In this chapter, we give an overview of the basics of AM. We explain the basic working principle of AM machines, briefly describe the most common AM technologies, the advantages and disadvantages of AM compared to conventional manufacturing techniques and, based on this, their areas of application. In the second part of this chapter, we discuss the process chain of AM service providers. The analysis of the process chain and the corresponding sub-processes forms the basis our work.

2.1 Additive Manufacturing

AM is defined as the group of production processes which produce components by the additive application of material. The first approaches of AM appeared in the early 1980's. Since then, many different AM technologies have been developed over the years: The most common techniques are Stereolithography (SLA), the PBF processes like SLS, Selective Laser Melting (SLM) or Electron Beam Melting (EBM), Fused Deposition Modeling (FDM), Laminated Object Manufacturing (LOM), Laser Metal Deposition (LMD) or Binder Jetting (BJG) [46, 38].

In this section, we will explain the basic working principle, the most common AM technologies, the advantages and disadvantages of AM over traditional manufacturing technologies and common applications for AM components.

2.1.1 Working Principle

Independent of the exact process of the respective technology, all AM technologies share some basic working principles. All processes are based on a digital CAD model and produce

the components using layer-wise application of material. The way from the digital CAD model to the final component can be divided into the following steps [33]:

1. CAD model generation: Regardless of the actual AM technology, a CAD model is necessary. This can be generated either by a CAD modeling software or e.g. by scanning of an existing component using a 3D-scanner.

2. Conversion to STL: Nearly all AM machines use the STL file format as standard. The STL file type describes 3D models using a surface representation, which is stored as a list of triangles. Each triangle is described by the three corresponding vertices and its normal vector [25]. Therefore, the CAD model which can be created using several different file formats, has to be converted to the STL file format.

3. Build job preparation: The size, orientation and position of the STL model on the AM machine has to be defined for the build process.

4. Data transfer and machine setup: After preparation, the STL model is transferred to the machine and several production parameters have to be set. An important parameter is the layer thickness. Based on the parameter layer thickness, the STL model is virtually sliced into horizontal layers. Therefore, this parameter affects the vertical resolution of the built components.

5. Additive Manufacturing: The component is produced layer by layer using the previously defined layer thickness. The actual production process depends on the AM technology which is used.

2.1.2 Technologies

AM technologies can be divided into four groups which are characterized by the base material they are using for the manufacturing process: Powder, liquid, filament and solid sheets [33, 46]. We will briefly explain all four groups. Most relevant for this thesis is the group of powder-based processes. The working principles explained here have an influence on the decisions made in Chapter 6 and 7.

Powder-based

The group of powder-based AM processes includes different varieties of AM processes which all have in common that they are using a heat source to selectively heat the production material in the form of powder to produce a component. Most common are the SLS process

for polymer processing and the SLM and EBM processes for metal processing. The SLM and EBM processes differ by the heat source they are using: SLM uses a laser while EBM is using an electron beam [46]. Since the basic working principles of these three technologies are relatively similar and the SLS process is most relevant for this thesis, we use it as an example process for the explanation of powder-based processes.

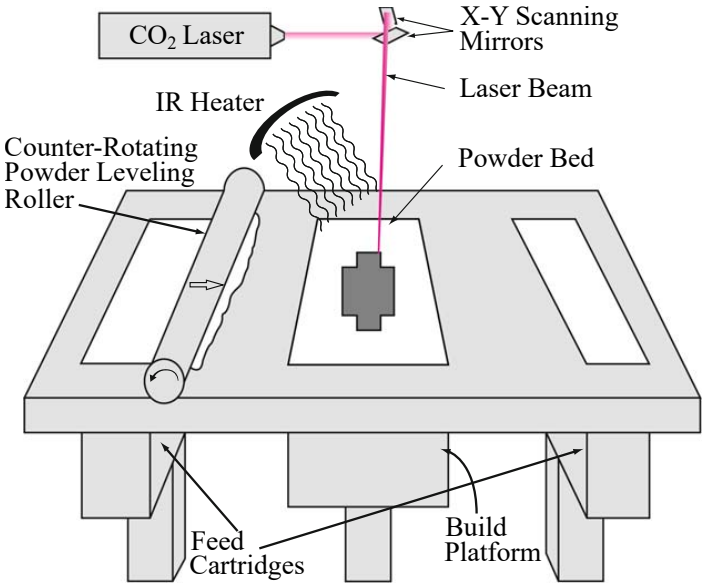


Fig. 2.1 The working principle of PBF machines [33].

The basic working principle of the production of a single component is shown in Figure 2.1. The layer-wise production is achieved by applying a layer of powder to the build platform using a powder leveling roller. The powder is provided by one or two feed cartridges on the left and right side of the build platform. The applied powder is heated up to a temperature just below the melting temperature of the used material using IR heaters. It can then be selectively sintered using the CO_2 laser [33].

Compared to melting, sintering is connecting adjacent powder particles without completely melting the powder [64]. With the scanning mirrors, the laser beam is controlled in a way that the current cross-section of the desired component is sintered. After this step, the build platform is lowered by one layer thickness and a new layer of powder is applied to produce the next cross-section of the component [33].

Using the basic process explained above, PBF machines can be used to produce not only a single component in one build process but as many as fit into the build chamber in parallel. Compared to SLM and EBM, SLS has an important difference. In the SLM and EBM machines, the produced components have to be produced with a fixed connection to the build platform. Therefore, multiple components can only be placed besides each other

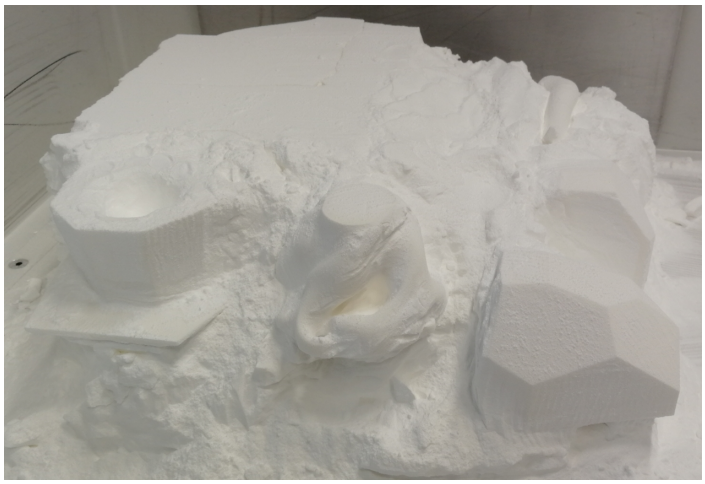


Fig. 2.2 Powder cake after SLS production

directly on the build platform. In contrast to that, in the SLS process components can be produced without connection to the build platform and can therefore be three dimensionally nested in the whole build chamber. In the build process, the components are supported by the powder between and beneath them what results in a so called powder cake (See Figure 2.2). Due to the three dimensional nesting, the build chamber can be optimally utilized during production.

The PBF process is used for various applications from prototyping to high quality end user components. The different PBF procedures offer the production of polyamide, metal or ceramic components with a resolution down to about 0.1 *mm* [33].

Filament-based

Filament-based AM machines like FDM machines generate components by extruding filament through a nozzle (See Figure 2.3). The FDM process starts with feeding filament material e.g. in form of a wire by a roller system into the liquifier chamber. Inside this chamber the material is liquified and afterwards extruded through the nozzle by applying pressure on the material. By moving the nozzle on a calculated path, a component can be manufactured layer by layer [33]. FDM is often used for prototypes since it is cheaper than the other AM technologies in most cases [49, 144]. Therefore, there is a large amount of FDM machines used for the private market. A drawback of FDM is the relatively large resolution. Typical FDM machines are using a layer thickness above 0.1 *mm* [58].

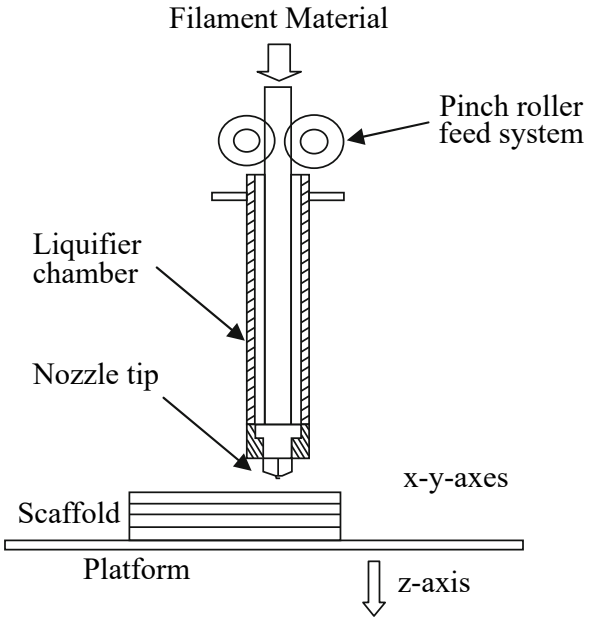


Fig. 2.3 The working principle of FDM machines [33].

Liquid-based

Liquid-based AM processes like SLA produce components using a liquid resin which is selectively solidified by photo-polymerisation [84]. The working principle of the SLA process is shown in Figure 2.4. The liquid resin is selectively solidified using an Ultra Violet (UV) laser. By lowering the base plate on which the first layer is adhering, the component can be produced layer by layer. One important property of SLA is that it can be used for the production of extremely filigree parts. Using so-called Two-photon Stereolithography (TPS), parts with a resolution of 0.1 μm can be produced [98].

Solid Sheet-based

Solid Sheet-based AM techniques like LOM produce components by layer-wise bonding of cut sheets. The basic working principle is shown in Figure 2.5. Material is added layer-wise by the insertion of e.g. polymer-coated paper or other materials via rollers to the build platform. For each layer the current cross-section of the component to be produced is cut into the polymer-coated paper by a laser and subsequently bonded to the lower layers e.g by thermal bonding. The unused material is sliced into squares using a cross-hatch cutting operation [33]. Advantages of LOM are the possibility to produce large sized parts and the fast production speed. A disadvantage are the weak connections between the layers due to the bonding process [65].

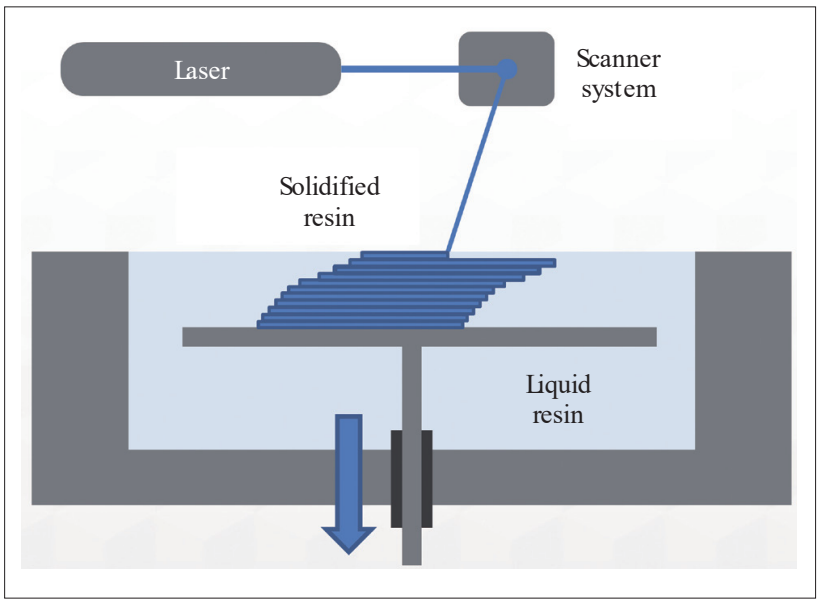


Fig. 2.4 The working principle of SLA machines [58].

2.1.3 Advantages

Due to its additive approach, AM offers several advantages compared to traditional manufacturing. A brief overview is given in the following subsections.

Production Costs for low Production Volumes

One main advantage is the decoupling of the production costs from the number of pieces to be produced. For the production of a component only a CAD model of the component is needed. Components can be produced without the necessity of any pre-processing steps like the production of a forming tool before the actual production. Compared to that, especially for complex geometries, often many different production stages are necessary to produce a component using traditional manufacturing techniques. If e.g. a mixture of molding and Computerized Numerical Control (CNC) machining is necessary, a mold for the molding and fixtures for the CNC machining would have to be produced prior to the actual production. These steps are not necessary using AM [33].

Production Speed

The properties which have a positive influence on the production costs also lead to a higher production speed for prototypes or small series. Without having the necessity of performing prior stages like mold construction, AM can be used for the rapid production of a component.

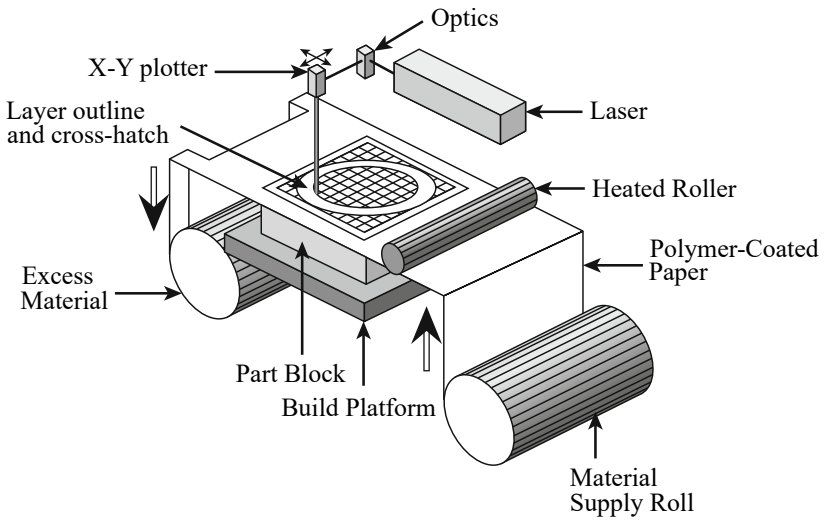


Fig. 2.5 The working principle of LOM machines [33].

This advantage can help reducing the time to market for new components by an acceleration of prototyping or enable the fast production of spare parts [11].

Design Freedom

Due to the independence of forming tools, AM offers the potential of high geometric design freedom. This design freedom can be used for the production of components which are optimized for a certain use case. Special characteristics of AM components can e.g. be lightweight or extremely complex parts. Additionally, complex parts which are produced using conventional manufacturing techniques often have to be manufactured by assembling various individual components. With AM, on the other hand, these components can often be manufactured as a single component, which allows better mechanical properties to be achieved [11].

Waste Reduction

A further benefit also results from the additive application of material. Compared to many traditional manufacturing techniques, AM produces less material waste. Especially subtractive production techniques produce waste material when removing material while producing a component. Some AM techniques, like the filament-based technique FDM, are nearly completely waste-free since only the necessity of adding support structures in the build-process need additional material. In PBF processes, where the components are produced inside the powder, at least parts of the unused powder can be recycled [70, 130].

2.1.4 Disadvantages

Besides the advantages explained in Section 2.1.3, AM also has some disadvantages compared to traditional manufacturing techniques.

Production Costs for high Production Volumes

In Section 2.1.3, we have explained that the independence of AM from forming tools leads to the independence of the production costs from the quantity of a component. Since this is an advantage for low production volumes, it changes more and more to a disadvantages with growing production volumes. Using AM, the price per part stays more or less constant. In contrast, for traditional manufacturing techniques, the price per part is decreasing with an increasing production quantity, since the impact of the fixed costs for a forming tool is decreasing. Therefore, using AM becomes more expensive than traditional manufacturing as soon as the number of pieces to be produced exceeds a threshold which varies depending on geometry, material and other factors [100].

Design Freedom

As mentioned in Section 2.1.3, the high geometric design freedom is a big advantage of AM. Nevertheless, the design freedom also leads to challenges. The design freedom is accompanied by a lack of concrete design rules. Especially for complex parts, it is often not clear which exact geometries are manufacturable and which are not. Therefore, a high expertise is necessary for being able to design 3D models which are optimized for AM [100].

2.1.5 Applications

The special characteristics explained in Section 2.1.3 lead to various use cases where AM parts have advantages over conventionally manufactured components. Additionally, AM offers the opportunity to be utilized by the innovative business model of AM service providers. These can use the advantages of AM to offer completely new manufacturing services.

Use Cases

As described in Section 2.1.3, AM offers the advantages that the production costs for a component are independent from the quantity to be produced and offers the possibility of producing extremely complex geometries. These advantages result in AM components being used primarily in industries with high technical demands like aerospace, automotive, biomedical and energy industry [46].

Especially the aerospace and automotive industry can highly benefit from the reduction of weight of the used components. Using lightweight components can e.g. decrease the operating costs of an airplane. For lighter airplanes, a weight reduction of one kilogram can lead to fuel savings of 1300 Dollar per year [11].

For the medical industry, AM offers several opportunities. It can e.g. be used to produce bone models using Computer Tomography (CT) scans of a patient. These can be used for the optimization of titanium mesh prostheses [33]. In addition, additive manufactured components can directly be used as implants e.g. for hip rod or stem implants manufactured as porous structures to support bone ingrowth [92].

Additionally, for many industries, it is attractive to integrate AM for their spare parts logistics. Often, spare parts for the company's own machines or sold products must either be available in the warehouse in sufficient quantities or the appropriate tools must be available for the production of the spare parts. This leads to unnecessary storage costs. With the help of AM, spare parts can be produced cheaply in the required quantity at short notice [57].

AM Service Providers

The advantages of AM lead to new business opportunities like AM service provision. Through the flexibility of AM machines, AM service providers are able to produce a wide variety of different parts using different materials with a relatively small machine park. Additionally, the independence from forming tools makes it possible to produce nearly any part while having relatively low costs. In combination with state-of-the-art Information Technology (IT) infrastructures and web platform-based online shops, it is possible to offer on demand AM services and produce and ship customer components within a few days.

Worldwide, more and more AM service providers are entering the market. Since the beginning of the millennium, the revenues of service providers worldwide increased from 200 millions to nearly 3000 millions of dollars in the year 2017 [23].

2.2 Process Chain of AM Service Providers

For the development of solutions for further automation of the processes in the process chain of AM service providers, we first have to analyze the process chain and the corresponding sub-processes. Therefore, in this section, an overview of the process chain of AM service providers is presented. We describe all processes that the 3D model respectively the corresponding component passes through on its way from data upload to packaging and dispatch. Of course, the process chain of AM service providers includes further steps besides those shown in Figure 2.6. However, these have been deliberately removed in our

graphic, as they are not directly affecting the production of a single component. Since the AM service provider we used as reference has its focus on PBF machines, we are focusing on that manufacturing technique, too.

In the context of this thesis, we are working on the further automation of the process chain of AM service providers using intelligent data analysis methods. Therefore, we are focusing on the sub-processes which are connected to the information flow of the digital 3D models which represent the central element of the whole process chain. Nevertheless, for the sake of completeness and for a better understanding of the process chain, we also include some mechanical process steps like sand-blasting or mechanical post-processing in our explanation of the process chain.

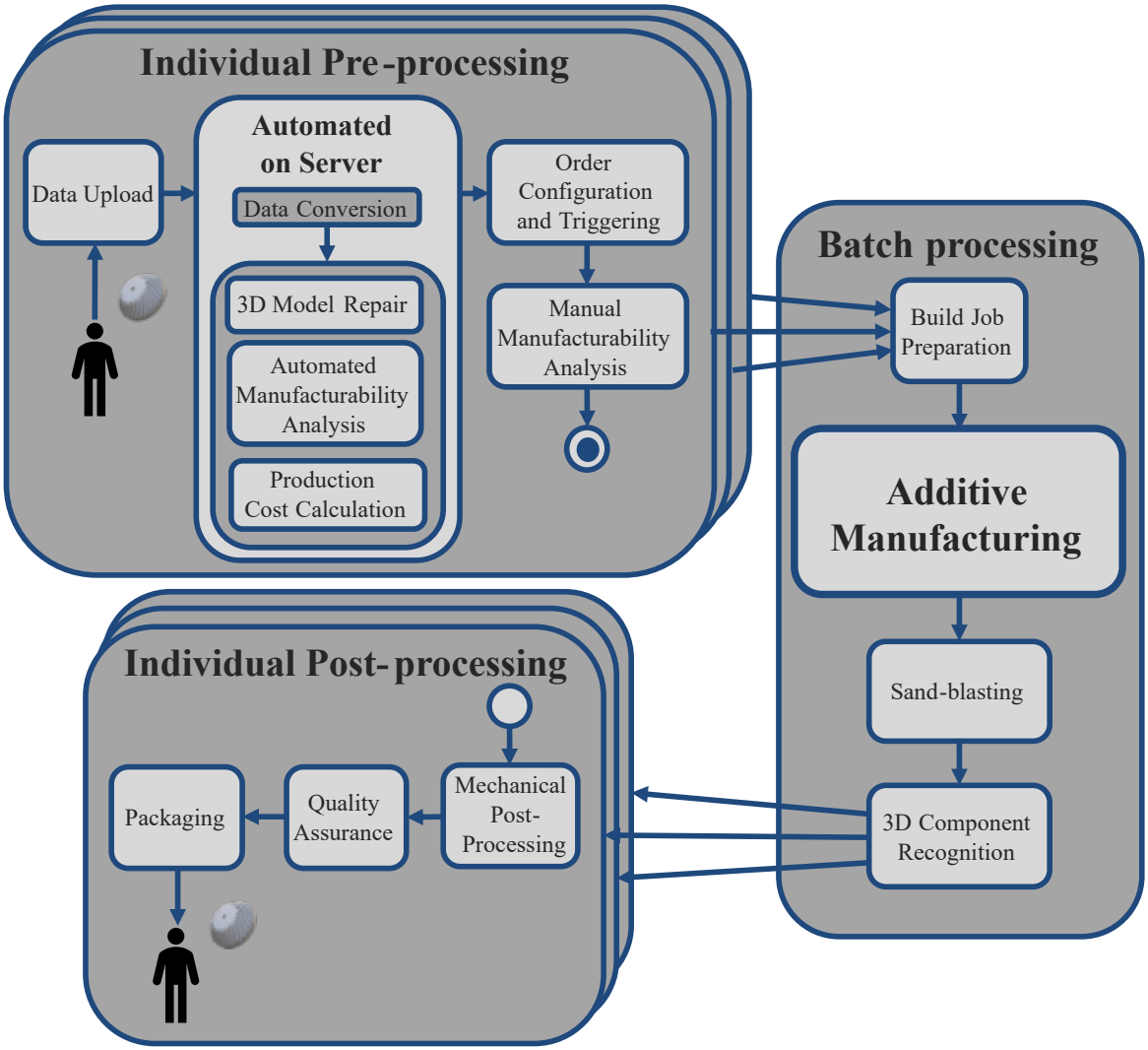


Fig. 2.6 The processes of AM service providers which are related to the digital 3D models [94].

Today's AM service providers are offering AM services for all potential customers from private persons to companies. For the production of a component, the corresponding digital 3D model and meta-information like material, color and quantity are necessary. Therefore, most AM service providers are using a web platform for the transmission of the data and the subsequent ordering process. The process chain description in Figure 2.6 therefore begins with the upload of the customers data and finishes with packaging and dispatch. As shown in Figure 2.6, the process chain is split into three main blocks.

2.2.1 Individual Pre-processing

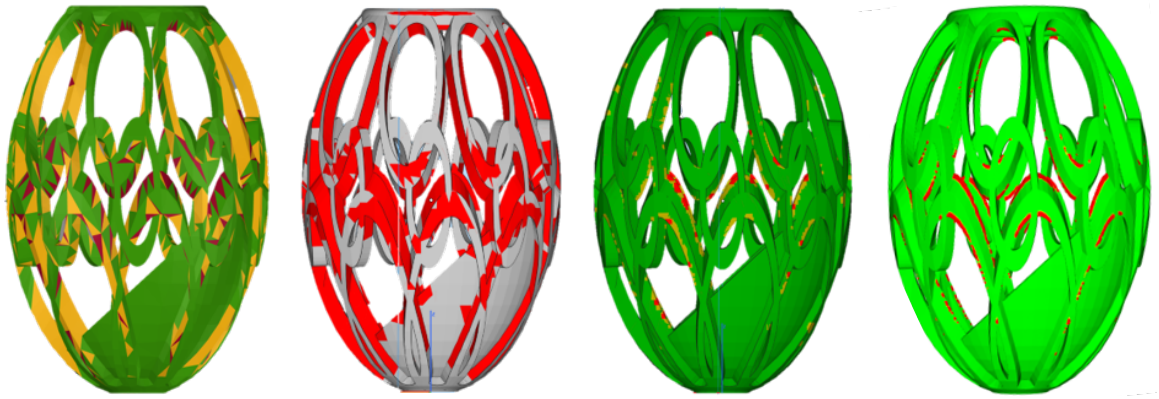
The first block of the process chain shown in Figure 2.6 includes several pre-processing steps which are carried out individually for each 3D model, starting with the data upload of a customer.

Data Conversion

Subsequent to the data upload, several data analysis steps must be executed for each 3D model. The first step is the data conversion step. Most digital 3D models are designed using a CAD modeling software or created by scanning a physical object using a 3D scanner. Since different CAD software tools and different 3D scanners are using different CAD data types, a standardized data type is necessary for being able to execute all further steps consistently. Therefore, all digital 3D models are converted to the STL file type prior to all following steps. STL files are triangle meshes stored as a list of triangles. For each triangle the three corresponding vertices and the normal vector are stored [25]. For the data conversion, again several different CAD processing tools exist. The conversion itself is executed within a few seconds depending on the complexity of the 3D model.

3D Model Repair

The data conversion step is followed by the 3D model repair. Errors can occur during the conversion process of the CAD models to the STL file type. Common errors are flipped normal vectors or the erroneous creation of multiple shells inside the mesh. To guarantee that all subsequent steps can be performed correctly, these errors have to be fixed. The repair step is again executed within a few seconds.



a) b) c) d)

Fig. 2.7 Examples of the wall thickness analysis results of different tools: a) A tool used by Dick & Dick GmbH [26], b) Materialise Robot triangle-based [83], c) Materialise Robot volume-based [83] and d) a tool used by the Shapeways company [120].

Automated Manufacturability Analysis

The following two steps *Manufacturability Analysis* and production cost calculation can be executed in parallel. Using the different AM processes, a wide range of geometries can be manufactured. Nevertheless, not all geometries are manufacturable. Therefore, all 3D models have to be analyzed to verify their manufacturability. For that purpose, existing AM data processing tools offer a wall thickness analysis which is able to analyze if parts of a 3D models geometry are beneath a certain minimum wall thickness threshold. However, the wall thickness analysis is not used by all AM service providers, since the results of the various tools can often not achieve satisfactory quality. Figure 2.7 shows the results of the analysis of an example 3D model using different wall thickness analysis tools. Areas that are identified as being below the wall thickness threshold are marked in red or yellow. As can be seen, the tools a) used by Dick & Dick GmbH [26] and b), the triangle-based version of the Materialise Robot [83], are marking many areas which are incorrectly classified as being beneath the wall thickness threshold. The tools c), the volume-based version of the Materialise Robot [83] and d), a tool used by the Shapeways company [120], provide better quality, but still contain incorrectly classified areas. The tools are able to check this criterion within a few seconds.

The *minimum wall thickness* criterion is only one of many so called *design rules* developed by different researchers from the AM domain [1, 62, 109]. The other design rules can not be checked using automated software tools, yet.

Production Cost Calculation

Since most AM service providers want to offer direct ordering via the web platform without having the necessity to write a manual offer, an automated production cost calculation step is performed. The costs for manufacturing a component using AM directly depend on the 3D models geometry, the chosen material and the quantity of parts to be produced. Additionally, the costs are affected by pre-processes like the build job preparation step or post-processes like surface refinement, coating or quality assurance. Since the workload and the corresponding costs for that steps are influenced by the 3D models geometry, it is a non-trivial task to calculate the costs for the production of an AM part based on its geometry and the meta-information. Currently used calculations are manually designed on the basis of the information of previous production processes and calculate the expected costs for a component instead of the exact costs. The costs are calculated for all materials an AM service provider is offering and possible post-processing steps. Similar to the steps before, the production costs are calculated within a few seconds.

Order Configuration and Triggering

When the analysis steps are finished and the costs for a 3D model using different materials and post-processing steps are calculated, the customer can configure his component and trigger the order. When the order is triggered, it is added to a First in First out (FIFO) order queue which stores all currently unfinished orders. Depending on the time a customer places his order, it takes different amounts of time for it to be processed. If the order is e.g. triggered in the evening, it will be processed the next day. Therefore, an order normally stays in the order queue up to 16 hours.

Manual Manufacturability Analysis

Subsequent to the automated data analyses steps, a manual *Manufacturability Analysis* is performed. As explained above, if at all, the *minimum wall thickness* criterion is automatically checked. For other design rules, currently there are no software tools available which provide automated manufacturability checks. The AM engineers have to analyze each 3D model manually using their expertise in order to guarantee that a component can be produced without production failures. Depending on the complexity of a 3D models geometry, the analysis can take up to a few minutes.

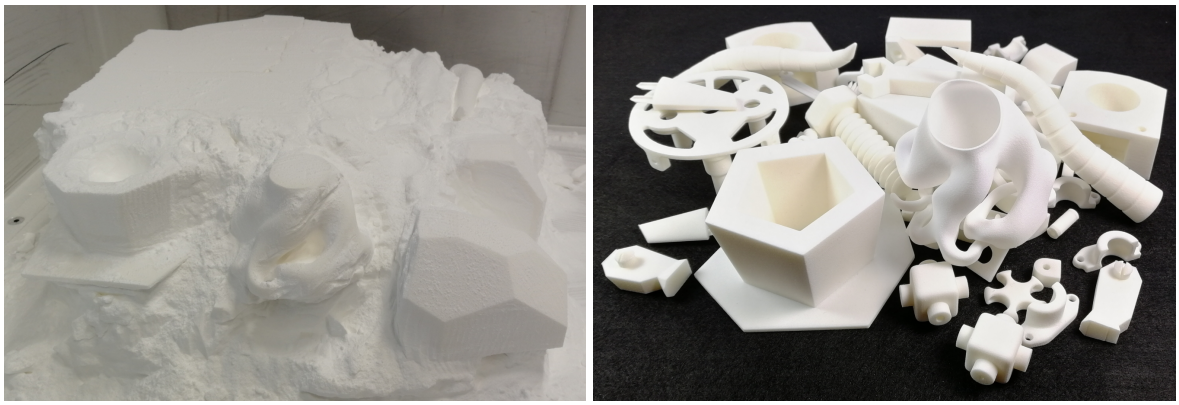
2.2.2 Batch Processing

All previous steps are carried out individually for each 3D model. For the following steps from build job preparation to *3D Component Recognition*, the 3D models are processed in batch processing mode. This means that a group of 3D models corresponding to one build job is processed simultaneously in the following steps.

Build Job Preparation

Fig. 2.8 Example for the digital preparation of a SLS build job.

AM machines can produce multiple components of the same material together in a single build job. These build jobs are manually prepared by an AM engineer using a build job preparation software. An example build job of the SLS technology is shown in Figure 2.8. The different 3D models are nested three dimensionally in the build chamber. For each 3D model, the AM engineer has to choose the orientation used for production. Additionally, he can choose the exact position in the build chamber. Alternatively, an automated tool can be used for the 3D nesting. In addition to the parameters mentioned above, the machine parameters like layer thickness, laser energy or laser movement speed can be set in the build job preparation step. The preparation of a single model only takes a few minuets at most. Nevertheless, the preparation of the whole build job can take more than an hour.



(a) Powder cake with components (b) Clean components after sand-blasting

Fig. 2.9 Components of a build job before (left) and after sand-blasting (right).

Additive Manufacturing

When a build job is completely prepared, the actual production process can be started. All processes executed by an AM machine are completely automated. Based on the prepared build job and the machine parameters, the machine is producing the components as explained in Section 2.1.2. Depending on the build chamber utilization and the machine used, the completion of one build job usually takes between 12 and 24 hours. In some cases, it can take up to 36 hours.

Sand-blasting

As shown in Figure 2.8, multiple components are manufactured simultaneously in one build job. As already explained in Section 2.1.2, especially using the SLS process, the components are manufactured without a fixed connection between each other. Therefore, when the production is finished, the different components are located besides each other in the powder cake (see Figure 2.9(left)). For preparing the components for the next process steps, they are cleaned via sand-blasting. For most build jobs, this is done using an automated sand-blasting station. Only for build jobs which include very filigree parts, the sand-blasting step is performed manually. The cleaned components after the sand-blasting step can be seen in Figure 2.9 (right). The duration of this step is again dependent on the utilization of the build chamber. Usually this step takes about 30 to 60 minutes.

3D Component Recognition

In the *3D Component Recognition* step, the digital data flow meets the physical process chain again. In order to determine the appropriate post-processing step for each component, these

must be assigned to the corresponding order again. Currently, employees are assigning the different components to the correct orders through a manually performed visual comparison of each component and all digital 3D models of the corresponding build job. The duration of this step is extremely dependent on the build chamber utilization and also the composition of different components in the build chamber. Especially for the SLS process, often up to 100 different components are produced simultaneously. On the one hand, for build jobs with few different parts, this step can be done within about 15 minutes. On the other hand, when a build job includes many different components, it can take more than an hour.

2.2.3 Individual Post-processing

When the components are assigned to the corresponding order, they are processed individually again.

Mechanical Post-processing

For each component, the post-processing steps which the customer has chosen are performed. Possible steps are e.g. coating or surface finishing. Depending on the actual process, this takes between 30 minutes and two hours.

Quality Assurance

After finishing all necessary post-processing steps, the quality of the components is verified. An employee performs a visual comparison of a chosen subset of the components corresponding to a build job with the corresponding 3D models. The dimensions of important geometric features of the components are measured. That can e.g. be the inner radius of a hole or the outer radius of a cylinder. For a whole build job, this process takes about 30 to 60 minutes.

Packaging and Dispatch

The process chain ends with the packing and dispatch step. As soon as all components of an order have passed the quality assurance, they can be packed and shipped to the customer. The packaging takes about 30 minutes to an hour per build job. The shipping itself takes about one or two days.

The time for the different processes is shown in Figure 2.10. The whole process takes between two and four days for the production of a polymer component using SLS. Depending on special configurations for a component it can sometimes take more time.

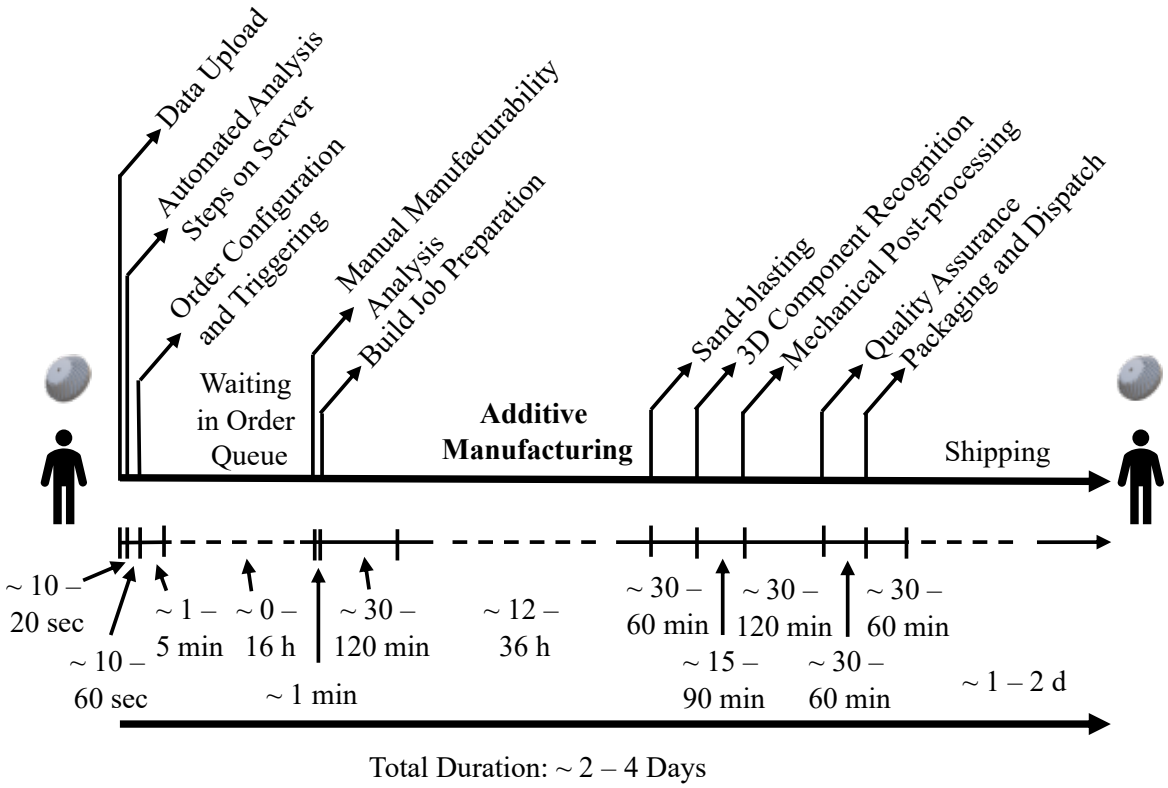


Fig. 2.10 The time necessary for the different sub-processes.

Within this thesis, we want to elaborate whether and to what extent flexible methods of the ML domain are able to contribute to further optimization and automation of the complex process chain. Since it is not possible to consider the entire process chain within the context of this thesis, we have decided to consider two sub-processes of the process chain. Based on our analysis of the individual process steps and consultations with the AM service provider, we have decided to proceed with the sub-processes *Manufacturability Analysis* and *3D Component Recognition*. The former is a completely virtual analysis task, the latter is characterized by the interconnection of the digital and physical process chain. Therefore, the two steps have different requirements and demand different solution concepts. In the following Chapter 3, we will explain the current state and the associated challenges of the two process steps in more detail.

Chapter 3

Problem Description

In the following chapters of this thesis, we deal with the two process steps *Manufacturability Analysis* and *3D Component Recognition*. Based on the previous descriptions, we will discuss the two processes in more detail. We analyze the current status of the two process steps and the influence on the entire process chain. Our goal is to analyze what optimization potential the two process steps offer and what requirements are necessary for optimization through automation. We will use this information in the further course of this thesis to evaluate the related work described in Chapter 5 and to develop the solutions described in Chapter 6 and 7.

3.1 Manufacturability Analysis

To understand why the *Manufacturability Analysis* in the domain of AM is a very complex problem that differs from the *Manufacturability Analysis* for other manufacturing technologies, we first describe the causes for limitations of manufacturability for AM and how the limitations are currently defined. Subsequently, we explain the current status and develop requirements for possible automation solutions.

3.1.1 Definition of Manufacturability

In order to engage in *Manufacturability Analysis*, we must first understand what causes constraints in AM and how the constraints can be described. As explained in Chapter 2, we are dealing with the group of PBF processes. The manufacturability constraints depend on the process and especially on the material used for the production. In the context of our work, we will focus on the SLS process which is used for the production of polymer components and the SLM process used for metal components.

Causes for Manufacturability Constraints

Manufacturability constraints have a wide variety of reasons which range from simple dimensional constraints defined e.g. by the size of the build chamber of an AM machine to constraints resulting from the complex thermal behavior of the production materials which are used. In this section, we will give an overview of basic types of constraints which are relevant for the SLS and SLM processes.

The first group of relatively simple constraints is caused directly by the structure of the AM machines described in Section 2.1.2. The constraint for the maximum size of a component is the simplest constraint. Since the machines are producing the components inside a build chamber, the dimensions of a component can not exceed the dimensions of this build chamber. Another constraint results from the usage of the laser for the selective melting or sintering of the powder. Since the laser has a defined diameter, it is not possible to produce filigree structures or walls which are smaller than the diameter of the laser spot. For the production of a stable component, even twice the size of the laser diameter is necessary. Both of these constraints are relevant for the SLS and SLM process.

The constraints of the first group can easily be described as a function of the dimensions of the build chamber respectively the diameter of the laser spot. However, other more complex constraints exist. When producing bores or other cavities like cooling channels, for example, it should be noted that the powder must be removed from these cavities after production. Depending on the geometry of the cavities, it may not be possible to remove the powder completely. This can lead to bores or cooling channels that do not provide the desired function. If cooling channels are blocked by powder, the required functionality is not given.

Additionally, especially in the SLM process, the cyclical process of local heating of the powder using the laser and subsequent cool down creates a complex thermal process. If the cross section of a component is varying significantly between adjacent layers thermal distortions can occur during this cyclical process. The unequal cross sections of the components in the individual layers result in an inhomogeneous cooling behavior during the cooling process that follows after melting. This can cause the upper layers of a component to bend while cooling down or even cause thermal cracks[90].

As can be seen, there are many different reasons for manufacturability constraints. Some of them are easy to describe, others are very complex and therefore not easy to define. Since it is essential to know the limitations of manufacturability in order to take advantage of the benefits of AM, several researchers have been working on mathematical definitions of the limitations in recent years. In the following section, we will give an overview about existing definitions.

Design Rules

In the last decade, several researchers have focused on the definition of *design rules* for AM [1, 2, 32, 62, 3, 80]. The rules serve mainly as guidelines for component designers. With the help of the guidelines, component designers are able to design new components directly AM-compatible. For being able to describe the constraints, basic geometries such as cuboids or cylinders are used, which can be defined by their geometric parameters. The different constraints from simple to highly complex are subsequently described based on the basic geometries or combinations of the individual geometries. For the definition of the design rules, the researchers have designed and produced test specimens of the basic geometries. Based on the results of the physical production of these specimens, thresholds can be defined for the geometric parameters describing the geometries, which determine whether they are manufacturable or not. An example of the test specimens can be seen in Figure 3.1. To illustrate the structure of the rules, we will show some example design rules corresponding to the previously described causes for constraints like *minimum wall thickness* or description of feasible and non-feasible bore geometries.

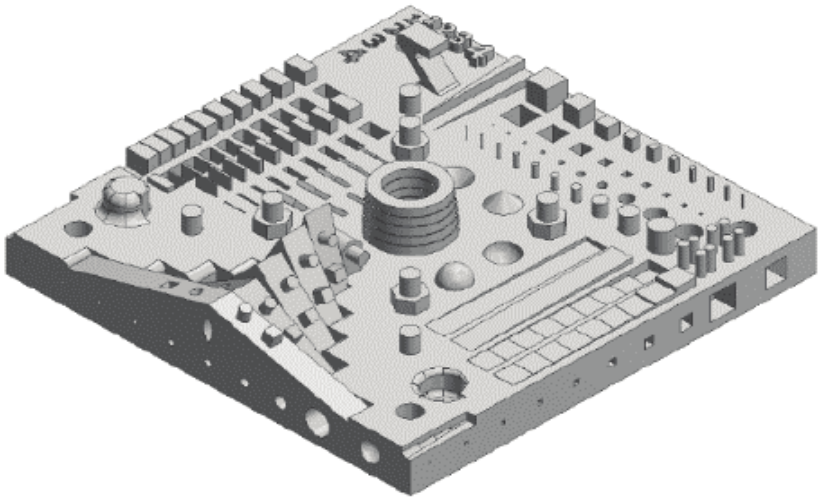


Fig. 3.1 Example of a test specimen. The different geometries like cuboids or cylinders are produced using different geometric parameters [37].

Figure 3.2 shows an example design rule for the *minimum wall thickness* criterion. As can be seen of the left hand side of the figure, this rule belongs to the basic element type of walls and defines the thickness criterion. As explained above, the *minimum wall thickness* that can be produced, depends on the diameter of the laser spot of an AM machine. For the creation of this design rule, Adam et al. [2] have not used the exact diameter of the laser spot

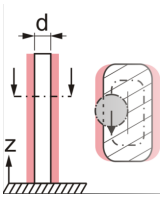
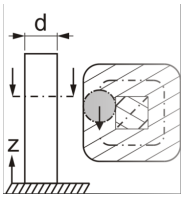
Group	Type	Attribute	Description	Design for manufacturing				
			Regular / Specific	Unsuitable design	Suitable design	LS	LM	FDM
Basic elements	Walls	Thickness	The thickness of a wall should be large enough to structure each layer with a boundary line and enclosed raster lines. *LS: d ≥ 1,0 mm* *LM: d ≥ 0,6 mm* *FDM: d ≥ 1,5 mm*			X	X	X

Fig. 3.2 An example design rule for the *minimum wall thickness* [2].

of a specific AM machine but standard values specifically for the groups of SLS, SLM or FDM machines.

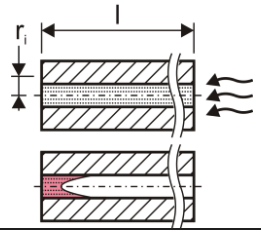
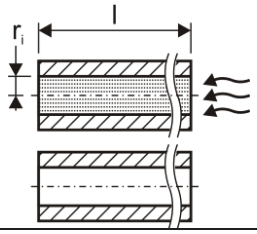
Bores' lengths should be short enough to enable a robust removal of powder materials from its insides.	
Laser sintering: BC_1.1: l ≤ 10 * r_i BC_2.1: l ≤ 10 * r_i BC_2.2: l ≤ 10 * r_i BC_2.3: l ≤ 8 * r_i	Laser melting: l ≤ 400 * r_i FDM: /
Unsuitable design	Suitable design
	

Fig. 3.3 An example design rule for the manufacturability of bores [3].

Figure 3.3 is showing a design rule for the manufacturability of bores. For the SLS process, it is necessary to maintain a certain ratio between the length l and the inner radius r_i of the bore. If the length is too large in comparison to the radius, the powder can not be removed from the bore. The variables *BC_x.x* describe different test sets used for the calculation of the ratio between length and radius [3].

Figure 3.4 shows two examples for design rules which describe the issue of thermal deformation. In both of the cases shown here, thermal deformations can occur in the SLM process. In the first case which is related to a transition of two basic elements, the cross section of the upper part A_3 must be smaller than or equal to the addition of the cross sections of the two lower elements A_1 and A_2 to ensure a faultless production. This means that

Extract from the design rules catalog.

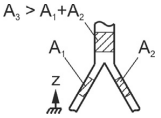
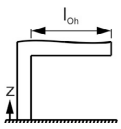
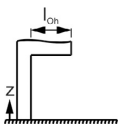
Group	Typ	Attribute	Description	Design for manufacturing		LS	LM	FDM
			Regular	Unsuitable	Suitable			
			Special					
Element transitions	Firmly bonded elements	Edge	Element transitions' thicknesses should be chosen so that the cross sectional areas in the building plane remain of the same size or become smaller.				X	
Aggregated structures	Overhang	Length	Overhangs' lengths should be short enough to ensure a robust manufacturability given by part layers that do not bent out of the building plane (LM) or filaments that do not "fall off" their nominal positions (FDM). LM: $l_{Oh} \leq 2.0\,mm$ FDM: $l_{Oh} \leq 1.8\,mm$				X	X

Fig. 3.4 Two example design rules which describe the problem of thermal deformation for the SLM process and the problem of material "falling off" for the FDM process (adapted from [1]).

the aspect ratio of the cross-sections should not increase. The second design rule defines constraints for horizontal overhangs for the SLM and FDM process. The constraint is defined via a fixed threshold value $l_{oh} \leq 2.0\ mm$ for the SLM process. This value is selected to be valid for the currently common SLM machines.

Depending on the exact AM technology, the various authors [1, 2, 32, 62, 3] have defined around 50 design rules, all of which are similar in structure to the examples already described above. The rules describe restrictions that either influence the quality of a component or even lead to production errors. If a component is designed to satisfy all design rules, it should be possible to produce it without errors.

3.1.2 Current State and emerging issues

As explained in Section 2.2, AM service providers need to verify the manufacturability of the components which their customers are uploading. This means that already existing 3D models have to be verified regarding their manufacturability with a certain AM method. The arising problem is that the design rules are not designed to evaluate existing 3D models but to serve as a guideline for the design of new components. As shown in Figure 2.6, the *Manufacturability Analysis* step is split into an automated and a manual analysis step. Currently, if at all, the automated tools which are on the market are offering a *Manufacturability Analysis* functionality which is able to check the *minimum wall thickness* criterion. All other criteria that may constrain the manufacturability of a certain 3D model, have to be checked by an AM engineer in the manual *Manufacturability Analysis* step. This leads to negative effects regarding production time and costs.

The whole work flow of the process chain of AM service providers in its current state can therefore be strongly influenced by the manual *Manufacturability Analysis*. The overall process time is heavily affected if a problem in a 3D models geometry is detected in the manual analysis step which was not detected in the automatic analysis step. The problem here is not the actual time for the manual analysis but the resulting delays. As can be seen in Figure 2.10, before an order and the corresponding 3D models are analyzed manually, they are waiting in a queue for being analyzed. If an issue regarding the manufacturability of a 3D model is detected, a manual consultation with the customer must be made and the customer may have to revise the component and re-upload it again. That often leads to an additional delay of one or more days. If the manufacturability issue would have been detected directly in the automated *Manufacturability Analysis*, the customer could have saved the time his order was waiting in the order queue. Therefore, a comprehensive automated *Manufacturability Analysis* solution would prevent unnecessary waiting in the order queue and thereby reduce the overall average process time. This would enable AM service providers to guarantee shorter process times from the upload of a 3D model by a customer to the delivery of the finished component.

3.1.3 Solution Requirements

As explained above, the automated *Manufacturability Analysis* in its current state has various problems. In the context of our work, we want to design a solution concept that extends or replaces the currently only partially functioning solutions. For being able to design a suitable solution, we first create a set of requirements based on the experience we made with the existing solutions and the findings generated with the analysis of the process chain. In order to be able to refer to the requirements in the further course of this thesis, we define them as R_{i-ma}. The index *ma* is marking that the requirement is related to the *Manufacturability Analysis*. For the process step of *3D Component Recognition*, we will define separated requirements marked with the index *cr*.

In Section 3.1.1 we have explained, that several different causes for manufacturability constraints exist but the current solutions can only evaluate parts of them. Therefore, we need a solution which is able to evaluate 3D models with regard to all types of constraints. Since the manufacturability constraints often change with the development of new AM machines, a solution has to be adaptable to new constraints. The first requirement for our solution is therefore defined as $R_{1-ma} \mathrel{\widehat{=}} adaptability$.

R_{1-ma} $\widehat{=}$ adaptability

Solutions must be designed to be adaptable. In addition to the variety of different manufacturability criteria currently existing, emerging criteria should also be easily integrable.

Besides that, a solution has to be able to evaluate the manufacturability criteria with a high $R_{2-ma} \widehat{=} accuracy$.

R_{2-ma} $\widehat{=}$ accuracy

The evaluation of 3D models in terms of their manufacturability must be performed with a high accuracy.

To achieve this, a suitable data representation must be used for the solution. The third criterion is therefore defined as $R_{3-ma} \widehat{=} suitable\ data\ representation$.

R_{3-ma} $\widehat{=}$ suitable data representation

In order to evaluate the 3D models with respect to the different manufacturability criteria, a data representation is required that contains all the necessary information with an appropriate level of detail.

Besides these criteria, which concern the pure detection of constraints, there are further criteria concerning the usability for the customers. A purely binary feedback whether a 3D model is manufacturable or not is only of limited help for the customers. If possible, customers need information on how they can adapt the geometry of their 3D model to make it manufacturable. The solution approach must therefore be able to generate *feedback* regarding critical geometries. Since it is necessary to mark these critical geometries within the 3D models, visual feedback is the most useful feedback option in our opinion. Therefore, the fourth requirement is defined as $R_{4-ma} \widehat{=} visual\ feedback$. This should provide as much information as possible.

R_{4-ma} $\widehat{=}$ visual feedback

Using visual feedback, critical part geometries are to be marked in a 3D model to provide customers with the necessary information to revise the 3D model.

Another important aspect of the ordering process of AM service providers is that it should be as simple and fast as possible. The analysis should therefore not last longer than the

Table 3.1 Ranking of the currently used industrial tools regarding the five metrics adaptability, accuracy, suitable data representation, feedback and computation time.

Method/Metric	R_{1-ma} Adaptability	R_{2-ma} Accuracy	R_{3-ma} Suitable Data Representation	R_{4-ma} Visual Feedback	R_{5-ma} Computation Time
Industrial Tools	– –	0	0	+	0

manual configuration steps the customer has to perform for a component. The configuration takes a few minutes at most. Therefore, the analysis should also be completed within this time frame. The last requirements is therefore defined as $R_{5-ma} \mathrel{\widehat{=}} computation\ time$.

$\mathbf{R_{5-ma}} \mathrel{\widehat{=}}$ computation time

The solution's computation time must not cause a delay in the customer's ordering process.

In the following, we want to rate the current industrial tools with regard to the defined metrics. Based on that ranking, we are able to identify strengths and weaknesses of the current solution. This information can be used for the development of our own solution. The ranking regarding the explained metrics for the current industrial tools which provide a wall thickness analysis is shown in Table 3.1. As explained in Section 2.2, several different tools exist for this purpose. Since the tools used by Dick & Dick GmbH [26] and the triangle-based version of the Materialise Robot are not providing a suitable quality, these are discarded. In Table 3.1, we summarize the volume-based version of the Materialise Robot [83] and the tool used by the Shapeways company [120] as *industrial tools*, since they offer a comparable performance. We will rate the performance with regard to the five requirements which we defined.

The major issue of the current tools is that they are not adaptable to further manufacturability constraints. They are only designed to verify the *minimum wall thickness* criterion. Therefore, we rate the performance regarding the requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$ as insufficient.

The tools are able to detect geometries with wall thicknesses under a certain threshold with moderate *accuracy*. The analyses of the tool currently used at the AM service provider have shown that the evaluations of the 3D models are rotation-variant. This means that if a 3D model is processed in different orientations, it may be evaluated differently. Therefore, we rate the performance of the industrial tools regarding the requirement $R_{2-ma} \mathrel{\widehat{=}} accuracy$ as fair.

The tools check the wall thickness directly based on the triangle information. Depending on the resolution of the triangles, this can lead to the check becoming very inaccurate if the resolution is too coarse or taking a long computation time if the resolution is very high. Therefore, the current data representation is not optimal and the performance of the solutions regarding the requirement $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$ is rated as fair, too.

The quality of the feedback also depends on the resolution of the 3D model which is evaluated. If the resolution is good, also the visual feedback is good. Nevertheless, when the resolution is too coarse, the quality of the visual feedback is decreasing. Therefore, we rate the performance regarding the requirement $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ as good.

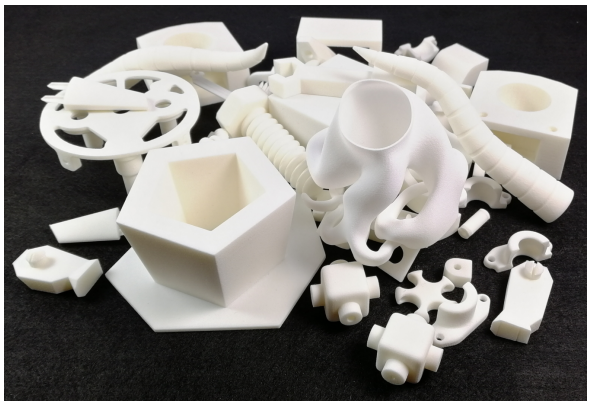
The computation time for most 3D models is in the range of a few seconds, for more complex 3D models which have a very high number of triangles, the analysis time can be significantly longer. The performance regarding the metric $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ is therefore also rated as fair.

The current state of the *Manufacturability Analysis* process clearly offers potentials for optimization. Overall, especially due to the lack of adaptability, the current solutions do not offer sufficient quality. This is also indicated by the fact that many AM service providers currently refrain from using the existing tools. Therefore, our goal is to develop an own solution that is optimized with respect to the described requirements.

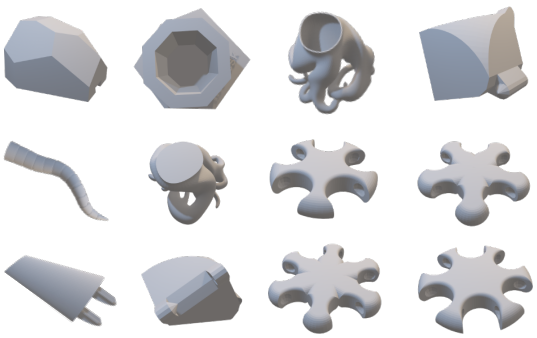
In Chapter 4, we will explain approaches from the field of ML which offer the potential for being used for designing an own solution for the issue of *Manufacturability Analysis*. In the following Chapter 5, we give an overview of innovative approaches to *Manufacturability Analysis* in other sectors of the manufacturing industry. Based on the information of these two chapters and the developed requirements shown above, we have designed our own solution architecture adapted to the problem of *Manufacturability Analysis* in the AM domain which we will describe in Chapter 6.

3.2 3D Component Recognition

As described in Section 2.2, it is necessary to recognize the produced components and assign them to the corresponding order because they are manufactured in batches which include multiple different components. Particularly for the SLS process, a build job can easily include up to 100 different components. This process step *3D Component Recognition* is currently performed completely manually. Since it is a time and therefore also cost consuming process step, this step offers great potential for optimization. In this section, we therefore address the issues created by the current practice of manual sorting, discuss whether the step could be optimized and define the requirements for possible optimization solutions.



(a) Clean components produced via SLS after sand-blasting.



(b) Sub-set of 3D models corresponding to the components of a build job.

Fig. 3.5 Components of an SLS build job after sand-blasting (left) and a sub-set of the corresponding 3D models.

3.2.1 Current State and emerging issues

As described in Section 2.2, the *3D Component Recognition* step has to be performed subsequent to the sand-blasting step. Since the sand-blasting is executed simultaneously for all components of one build job, the whole set of components corresponding to the build job is located together in a container after sand-blasting. Depending on the AM machine, each build job can contain up to several hundreds of components.

The *3D Component Recognition* is currently executed using manual visual comparison of the components (Figure 3.5 (left)) and the set of 3D models belonging to the corresponding build job (Figure 3.5 (right)). Each produced component must be compared manually with all 3D models belonging to the build job (See Figure 3.6). This is a cumbersome process which is time and therefore also cost consuming and error-prone.

As can be seen in Figure 2.10, the step takes between roundabout 15 and 90 minutes per build job, depending on the amount of different components and the overall mount of components produced within a build job. A major difficulty of this manual process is that employees can not adjust to a consistent process due to the daily variations of the process. The daily changing workload leads to a work flow which is difficult to plan exactly e.g. with regard to the amount of employees which will be needed to complete the task of *3D Component Recognition* on a given day. This may affect the duration of the entire process chain. Therefore, depending on the utilization of the AM machines, the duration of the whole AM process chain can fluctuate. It may happen that the recognition can not be completed for all produced components and thus the delivery of customer orders may be delayed by a day.

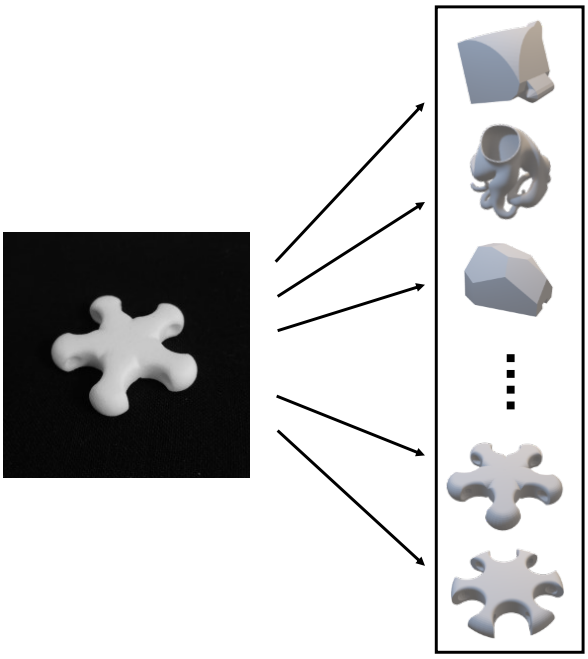


Fig. 3.6 Visual matching of a component and the 3D models corresponding to the build job.

Especially in consideration of the steadily increasing production volumes, the current approach is not practicable on a long-term basis. Therefore, an alternative solution is needed to replace the current manual approach with an automated solution. In the following section, we describe the requirements for possible alternative solutions.

3.2.2 Solution Requirements

In the long term, manual assignment of components will no longer be feasible. Therefore, we need an automated solution which ensures a consistent and plannable process. To ensure that all components of a daily production can be assigned to the correct customer order, a number of requirements arise for possible solutions. Similar to the requirements for the *Manufacturable Analysis*, we are defining the set of requirements R_{i-cr} in this section. The index *cr* is indicating that the requirements are related to the *3D Component Recognition* process.

The most important requirement is that all components which have been produced, can be assigned to their respective order in a fully automated manner, if possible. This must be achieved regardless of possible changes in the production environment due to, for example, contamination or abrasion. Thus, the most important requirements are to achieve a high recognition rate ($R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$) and a robust system $R_{2-cr} \mathrel{\widehat{=}} robustness$.

R_{1-cr} $\widehat{=}$ recognition rate

For a fully automated recognition of the components, a recognition rate of 100% is necessary, if this is possible.

R_{2-cr} $\widehat{=}$ robustness

The recognition rate should not be affected by external influences such as contamination of the sensors used for the recognition.

The next requirement is the need for adaptability of a possible solution ($R_{3-cr} \mathrel{\widehat{=}} adaptability$). As already described in Section 2.1.5, the composition of components to be produced is varying each day. Often completely unknown components are manufactured which have never been manufactured before. Therefore, a potential solution must be able to adapt to the daily changing production in a fully automated manner. A daily manual adjustment of the solution for the recognition of the varying components is not feasible.

R_{3-cr} $\widehat{=}$ adaptability

The solution must self-adjust daily for the recognition of the daily varying components to enable an optimal recognition rate.

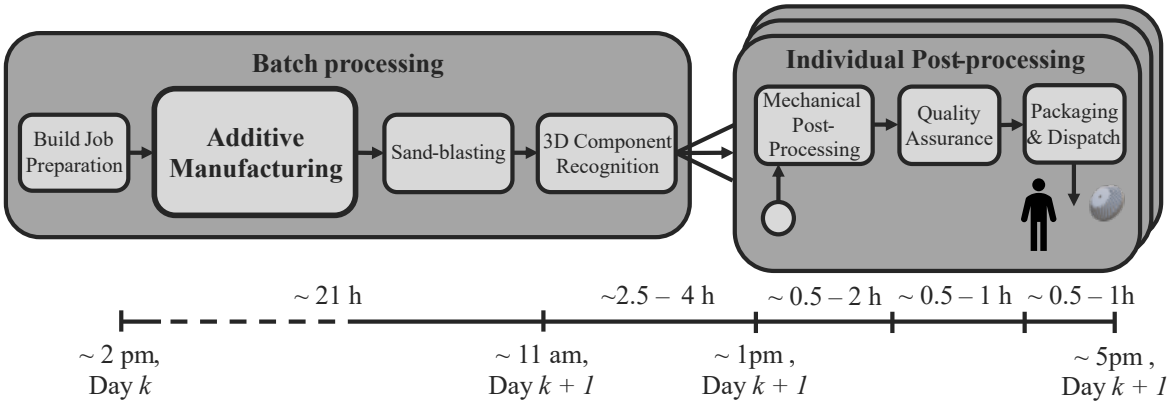


Fig. 3.7 The part of the process chain of the AM service providers which is influencing the requirements for possible solutions for automated *3D Component Recognition*.

In addition to these requirements, time constraints have to be considered. As fast production and delivery of the components is one of the key features of AM service providers, it must be guaranteed that the entire process chain can be run through without unnecessary delays. Regarding the process step of *3D Component Recognition*, this means that each component has to be assigned to the respective order and subsequently processed further

on the same day on which it is produced. The time constraints are defined by the overall process chain of AM service providers. We analyzed the process chain of the AM service provider, which we used as reference. Therefore, all time specifications mentioned here, refer to the process chain of that service provider. The process chain of other service providers can differ slightly but in principle includes the same steps. In Figure 3.7, we investigate the parts of the process chain that have an impact on the time constraint. The excerpt from the process chain shows the standard procedure from build job preparation to packaging and subsequent dispatch. The build job preparation is usually completed against 2 pm at an arbitrary day k. At this stage, it is determined which components are produced and therefore have to be recognized subsequent to the production at day $k+1$. A standard build job is tailored to complete production early in the morning at day $k+1$. At 11 am the sand-blasting step is usually completed and the components are prepared for the recognition step. To determine how much time is available for the *3D Component Recognition*, the remaining steps of the process chain must be considered. The process chain ends with the shipping of the components. This takes place at about 5 pm at day $k+1$. Most of the mechanical post-processing steps, quality assurance and packaging take between two and four hours in total. The recognition of the components must therefore be completed by about 1 pm at day $k+1$ to ensure that all further steps up to shipping can be performed in time. This implies that a period of 2.5 to 4 hours is available for the recognition of the components. This leads to the requirement $R_{4-cr} \mathrel{\widehat{=}} process\ time$ which must be suitable for the defined time periods.

R_{4-cr} $\widehat{=}$ process time

The solution must be able to recognize all produced components without causing delays in the overall process chain of AM service providers.

In this period, however, not only the components of a single build job but all components produced on that day must be recognized. Depending on the size of the machine park, this can be several thousand components. A possible solution must therefore be scalable so that it can be adapted to the amount of components produced ($R_{5-cr} \mathrel{\widehat{=}} scalability$).

R_{5-cr} $\widehat{=}$ scalability

The solution must be scalable to increasing production volumes.

Currently, high labor costs are also incurred due to the manual execution of the *3D Component Recognition* step. Therefore, another goal of the automation is to reduce the costs of this process step and thus the costs for the production of a component ($R_{6-cr} \mathrel{\widehat{=}} process\ costs$).

Table 3.2 Ranking of the manual process regarding the six metrics recognition rate, robustness, process time, process costs, scalability and adaptability.

Method/Metric	R_{1-cr} Recognition Rate	R_{2-cr} Robustness	R_{3-cr} Adaptability	R_{4-cr} Process Time	R_{5-cr} Scalability	R_{6-cr} Process Costs
Manual process	++	++	+	−−	−	−−

$\mathbf{R_{6-cr}} \mathrel{\widehat{=}}$ process costs

The solution should be designed to be as cost effective as possible to reduce the cost of the *3D Component Recognition* compared to the manual approach.

In Table 3.2, we evaluated the current manual approach with respect to the six criteria $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{3-cr} \mathrel{\widehat{=}} adaptability$, $R_{4-cr} \mathrel{\widehat{=}} process\ time$, $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$ which we described above. The current approach fulfills its purpose of a robust recognition of the produced components with a very high accuracy. Only individual errors can lead to incorrect assignments. Therefore, we rate the $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$ and the $R_{2-cr} \mathrel{\widehat{=}} robustness$ as excellent. The adaptability is given, too, since the employees are in principle able to recognize and assign the daily different components. However, the varying components make the work step significantly more demanding, as a high level of concentration is required. We therefore rate the performance regarding the requirement $R_{3-cr} \mathrel{\widehat{=}} adaptability$ as good.

The main problems, however, are the high process time required and the associated labor costs. The manual recognition of large amounts of manufactured components is extremely time consuming and therefore also expensive. The performance of the manual process with regard to the requirements $R_{4-cr} \mathrel{\widehat{=}} process\ time$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$ is therefore rated as insufficient. In addition, the manual approach can only be scaled by increasing the amount of employees. Therefore, the requirement $R_{5-cr} \mathrel{\widehat{=}} scalability$ is rated as inadequate.

Based on our analysis, we see a high potential especially for time and cost savings in the process step of *3D Component Recognition* by developing an automated solution which fulfills the requirements described in this section. Especially assuming that the overall production volumes of AM service providers will continue growing as they did the last years [145], the impact of an automation will grow with the rising volumes.

Since the digital data process chain and the physical process chain meet again in this process step, the step offers the possibility to use the digital information to optimize the physical process. Modern data processing methods offer the possibility to link the physical

and the digital process for the assignment of physical components to their digital twins. In Chapter 4, we will explain approaches from the field of ML respectively DL which can potentially be utilized for the creation of an automated solution for the *3D Component Recognition*. Subsequently, in Chapter 5, we give an overview of existing solutions in the AM domain and solutions for similar problems in related domains. In Chapter 7, we will describe our own solution for an automated *3D Component Recognition* which is based on the information described in the Chapters 4 and 5.

Part III

Foundations and Related Work

Chapter 4

Foundations of Machine Learning and Computer Vision

For the development of solutions for an optimization of the two sub-processes *Manufacturability Analysis* and *3D Component Recognition*, methods from the field of ML respectively DL play a decisive role. In Chapter 3, we have identified that for both tasks, the ability to adapt is of major importance for possible solutions. This adaptability is one of the greatest strengths of ML and DL methods. Based on data corresponding to a problem, ML and DL approaches are able to solve many problems, i.e. to adapt to them. However, depending on the actual task, domain, data availability and other influences, this adaptation process is not always a process easy to realize. Within the scope of this chapter, we therefore describe different methods that are essential for being able to develop ML resp. DL solutions that are optimally adapted to a task and the given circumstances.

In this chapter, we will therefore provide an overview about the underlying methods. Since this thesis is very interdisciplinary in nature, we first give a brief overview of basic ML methods and concepts in Section 4.1. These explanations enable readers with little or no prior knowledge regarding ML to get a better understanding of the rest of this chapter and the thesis.

In the Sections 4.1.5, 4.1.6 and 4.1.7, we explain different methods which can be used for tasks where no or only few data is available for solving the tasks. This is the case for the two sub-processes *Manufacturability Analysis* and *3D Component Recognition* we are dealing with.

Subsequently, in Section 4.2 the domain of DL will be introduced. Again, we provide a brief overview of the underlying methods for readers which are unfamiliar with the domain of DL. For the development of solutions for the *Manufacturability Analysis* and *3D Component Recognition* we need algorithms which are able to process 3D data. Therefore, in Section

4.2.2, we explain the basic approach of CNNs, which are Deep Neural Networks (DNNs) developed for processing grid-structured data, that means for example images or 3D data represented as grid structure. On the basis of the methods of transfer learning for ML described in Section 4.1.5, we describe the corresponding methods for DL in Section 4.2.3. These are essential for the development of our solution for the *3D Component Recognition*.

In Section 4.3, we introduce the field of XAI. As already stated in Section 3.1.3, a solution for the sub-process *Manufacturability Analysis* has to be able to generate feedback, which explains the decisions of the solution. Therefore, in the Sections 4.3.1 and 4.3.2, we first give an overview of basic methods for XAI. These again serve for better understanding for readers who are not familiar with this domain. Subsequently, in Section 4.3.3, we describe the concrete methods for XAI for DL which we are going to use for our solution.

The last section of this chapter deals with the field of CV. Both sub-processes, the *Manufacturability Analysis* and the *3D Component Recognition*, can be assigned to this domain. Therefore, we give an overview of traditional methods for the treatment of issues corresponding to the CV domain and subsequently DL-based methods specialized for solving tasks from the CV domain. These are based on the methods described in Section 4.2.2 and form the central element of our solution.

The field of ML is an interdisciplinary domain with influences from many different research areas. Therefore, it is not the aim of this chapter to provide a complete and detailed explanation of all areas belonging to the field of ML but to explain the methods which are relevant for the following chapters of this thesis.

4.1 Machine Learning

For being able to understand the concept of the different ML and DL methods, we first need to clarify what ML actually means and how ML algorithms are fundamentally structured. The first four subsections of this section are mainly intended for readers with limited experience concerning the field of ML. We give a brief overview about the main learning techniques, *supervised*, *unsupervised*, *semi-supervised* and *reinforcement learning* and subsequently explain general ML methods.

Based on the fundamentals described in the Sections 4.1.1 to 4.1.4, we describe concrete methods that enable ML and DL solutions to be used even if no or little data is available to solve a certain problem statement in the Sections 4.1.5 to 4.1.7. Especially when working on the problem of *3D Component Recognition*, we were faced with the challenge that initially no corresponding data was available to train a solution system. With the help of the methods

described in the Sections 4.1.5 to 4.1.7, we are nevertheless able to develop and apply a data-driven solution to address the issue of *3D Component Recognition*.

4.1.1 What is Machine Learning?

ML is the generic term for the field of research that deals with *learning* algorithms. In 1959, Arthur Samuel described it as the "field of study that gives computers the ability to learn without being explicitly programmed" [91]. But what does *learning* mean in this case? According to Mitchell, learning is defined as follows: ¨A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [86].

This means that ML systems are able to learn how to solve a task or at least approach the solution based on experience. The necessary experience is usually available in the form of training data. The solution for a problem is learned exclusively by processing corresponding data and is not explicitly programmed. This approach offers the possibility to train computer programs to be able to solve a problem for which no static solution algorithm exist based on corresponding data [135].

Instead of programming an algorithm which directly solves a given task T, only the way the ML model is processing the training data has to be defined for enabling it to learn. Using this learning-based approach, a wide range of different problem types can be solved: Classification or regression tasks, transcription or machine translation, anomaly detection or imputation of missing values. Depending on the task T, a performance measure P is defined. This can e.g. be the *accuracy* or *error rate* in case of a classification task. As already mentioned, the experience E is generated through the processing of data sets [39].

Nowadays, there are a lot of problems for which the effort for manual programming of a solution is so complex that programming is no longer reasonable. Some problems, due to their complexity, cannot even be captured by manually defined solutions at all. However, for many of these problems, ML models can be used to solve them. In the course of this chapter, we give an overview of the different types of approaches existing in the ML domain.

4.1.2 Types of Machine Learning

An important characteristic to distinguish different ML types is the way they learn. There are four large groups of learning methods: *supervised*, *unsupervised*, *semi-supervised* and *reinforcement learning*. The groups differ in the type of data which is used for their training. We will give a brief overview about the different types and subsequently go into more detail for the group of supervised learning techniques.

Supervised Learning

Supervised learning algorithms use data sets which consist of instances of input data vectors $\mathbf{x}$ and a corresponding label as value or vector $\mathbf{y}$. It is often used for e.g. classification or regression tasks if a sufficient amount of labeled data is available.

For the training of an ML model for the classification of emails into the categories *spam* or *no spam*, a set of emails and the corresponding label *spam* or *no spam* would be used for supervised learning. In this case, the emails would be the corresponding input data $\mathbf{x}$ and $\mathbf{y}$ the respective label *spam* or *no spam*. The supervised model is learning to predict $\mathbf{y}$ based on the input data $\mathbf{x}$ by estimating the conditional probability $p(\mathbf{y}|\mathbf{x})$ [39].

For a better understanding of the supervised learning procedure, we will briefly explain it on the example of linear regression. The linear regression algorithm is a simple method for solving a regression problem. The goal is to learn a linear function which calculates an output $\hat{y} \in \mathbb{R}$ based on an input vector $\mathbf{x} \in \mathbb{R}^n$. The linear function is defined as

$$\hat{y} = \mathbf{w}^T \mathbf{x} \tag{4.1}$$

where $\mathbf{w} \in \mathbb{R}^n$ is the vector of parameters called weights. These parameters can be trained using the training data. The goal of the training process is to adapt the parameters for minimizing a chosen error metric like the Mean Squared Error (MSE) on a chosen test set. The MSE is defined in Equation 4.2

$$MSE_{test} = \frac{1}{m} \sum_i (\hat{\mathbf{y}}^{(test)} - \mathbf{y}^{(test)})_i^2 \tag{4.2}$$

The marker $(test)$ indicates that a test data set which is disjunct from the training data is used. The test set contains m elements. The MSE thus describes the mean squared deviation of the calculated output values $\hat{\mathbf{y}}$ from the correct values $\mathbf{y}$ [39].

In the training process itself, not the MSE_{test} but the MSE_{train} based on the training data is minimized. If done correctly, also the MSE_{test} is minimized that way. MSE_{train} can be minimized in the training process by solving Equation 4.3.

$$\nabla_w MSE_{train} = \nabla_w \frac{1}{m} \sum_i (\hat{\mathbf{y}}^{(train)} - \mathbf{y}^{(train)})_i^2 = 0 \tag{4.3}$$

In other words, the weights $\mathbf{w}$ for which the gradient of the MSE_{train} is zero are searched for. This point is equivalent to the low point of the MSE function. This optimization of parameters in relation to a chosen metric is not only used for logistic regression but in general in the field of *supervised learning*.

Unsupervised Learning

In contrast to *supervised learning*, ML models trained using *unsupervised learning* only receive data vectors **x** without the corresponding labels **y** as training information. Instead of learning to predict an output label **y**, unsupervised learning systems are often used to find patterns in the underlying data set. Their capability can be used to create additional information about the data which is processed which can be used to get a better understanding of the data set. Classical tasks are clustering or dimensionality reduction [31].

Semi-supervised Learning

As the name suggests, *semi-supervised learning* models use both labeled and unlabeled data. In many domains, not enough data is available to perform a purely supervised training of an ML model for a certain task. Collecting or generating additional labeled data is usually expensive and time consuming. In such cases, semi-supervised learning is a good option. With the help of large amounts of unlabeled data, an unsupervised training can first be carried out, whereby the ML model can generate information that enables it to learn to solve a problem with a much smaller amount of labeled data in the second step [153].

Reinforcement Learning

In *reinforcement learning*, a system interacts with its environment. Thereby, it learns which actions a have to be performed to get a reward r. The procedure is based on the trial-and-error principle. In the learning process, the system is trying different actions a respectively series of actions a_1, a_2 .. a_n and learns which series of actions leads to the best series of rewards r_1, r_2, .. r_n. Since early actions a_i can also influence later rewards r_{i+k}, the system does not receive direct feedback regarding long-term effects but delayed feedback. A classic example problem for a reinforcement learning system consists in learning how to play chess [129, 31].

For the development of solutions for the two sub-processes *Manufacturability Analysis* and *3D Component Recognition*, labeled data for training *supervised learning* solutions is available or can be generated with the methods we explain in the Sections 4.1.5 to 4.1.7. Therefore, among the ML types described above, only the technique of *supervised learning* is relevant for the further work in the context of this thesis. In the next section, we will briefly explain the general structure of ML algorithms and subsequently provide an overview of different ML algorithms that can be used for *supervised learning* in Section 4.1.4.

4.1.3 Structure of ML Algorithms

Almost all ML algorithms consist of the building blocks of a data set, an ML model, a cost function and an optimization algorithm. In the case of *supervised learning* the data set consists of a set of input data instances **X** and the corresponding labels **y**. The ML model can be e.g. the linear regression model which we already described in Section 4.1.2 or any other ML or DL model. The cost function is used to describe the performance of the ML model and can be optimized using the optimization algorithms. These four building blocks have to be chosen with respect to a given problem [39].

4.1.4 Algorithms

In this section, we will give a brief overview about different ML algorithms including their basic working principle and applications for which the algorithms are used. The purpose of this section is to provide an overview of possible basic solutions for problems that are supposed to be solved by supervised learning methods. Since there is a large number of different algorithms, we will only deal with the most common approaches.

Logistic Regression

The Logistic regression is a popular algorithm for solving multivariable problems where an outcome D depends on a set of k independent input variables X_i with $i \in 1,..,k$ [59]. Even if the name suggests the opposite, *logistic regression* is mostly used for classification tasks [39]. The logistic function is defined as

$$f(z) = \frac{1}{1+e^{-z}} \tag{4.4}$$

where z is defined by a combination of the input variables. As can be seen in Equation 4.5, the influence of the different variables can be weighted via the parameters β_i and additionally a bias α is added. These parameters and thus the influence of the corresponding input variables can be learned in the training process [59].

$$z = \alpha + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k \tag{4.5}$$

In contrast to *linear regression*, for *logistic regression*, there is no closed-form solution for calculating the optimal values for the parameters. Therefore, for optimizing the parameters, the log-likelihood has to be maximized by minimizing the negative log-likelihood using the gradient descent approach. The gradient descent approach is an iterative approach for finding a local minimum in a function by an iterative movement downhill towards the minimum

using a small fixed step size. Gradient descent, respectively the stochastic gradient descent approach which is based on it, forms the basis for almost all learning procedures in the domain of supervised learning [39].

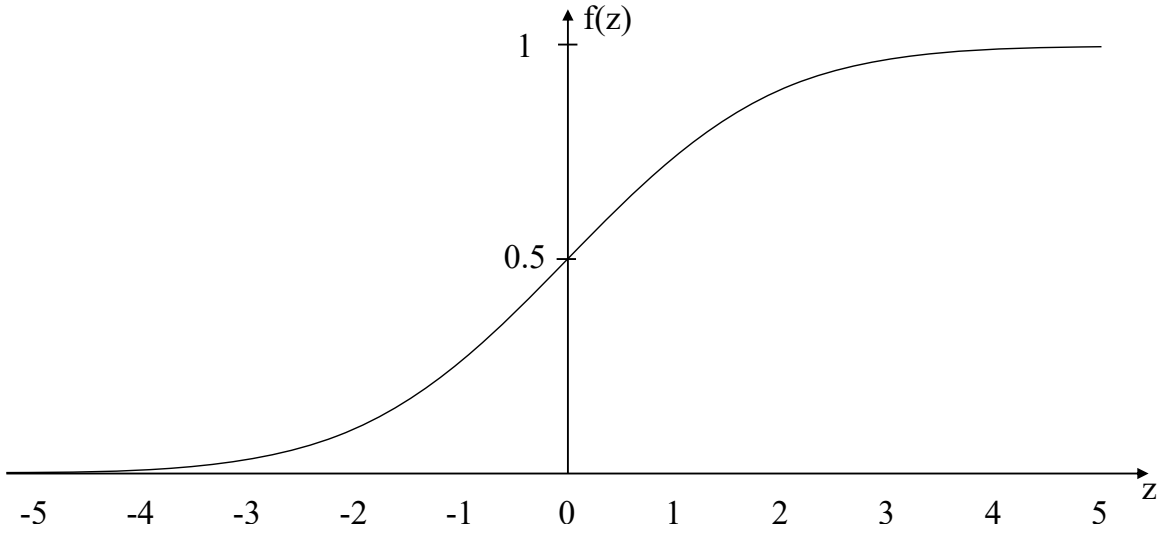


Fig. 4.1 The logistic function.

The resulting function is shown in Figure 4.1. A big advantage is, that the output of the logistic function ranges from 0 to 1 what makes it perfectly suited for describing a probability. Additionally, the S-shape of the function is well suited for describing several problems like epidemiological forecasts since the function saturates towards ∞ and $-\infty$ while being most sensitive to changes in the value range around zero [59].

k-nearest neighbors

Another popular algorithm is the *k-nearest neighbors* algorithm. In contrast to the linear and logistic regression described earlier in this chapter, the *k-nearest neighbors algorithm* is non-parametric and can be used both for classification and regression tasks. The output value y for an input vector $\mathbf{x}$ is calculated by finding the k-nearest neighbors of $\mathbf{x}$ and averaging the value for a regression task or using a one-hot encoding for a classification task [39].

Support Vector Machines

Support Vector Machines (SVMs) belong to the group of kernel machines. They are using the so called kernel trick to transform a non-linear problem into a higher dimensional space where the problem can be solved using a linear model. This linear model can then be

optimized again using convex optimization what is easier than trying to solve the non-linear problem in its original space.

Using this procedure e.g. for a binary classification task, the two classes can be separated by a hyper-plane in a high-dimensional space. This hyper-plane is calculated using the SVM approach. Figure 4.2 shows a 2D example for a classification using an SVM. The instances of the two classes are marked using dots and plus signs [5].

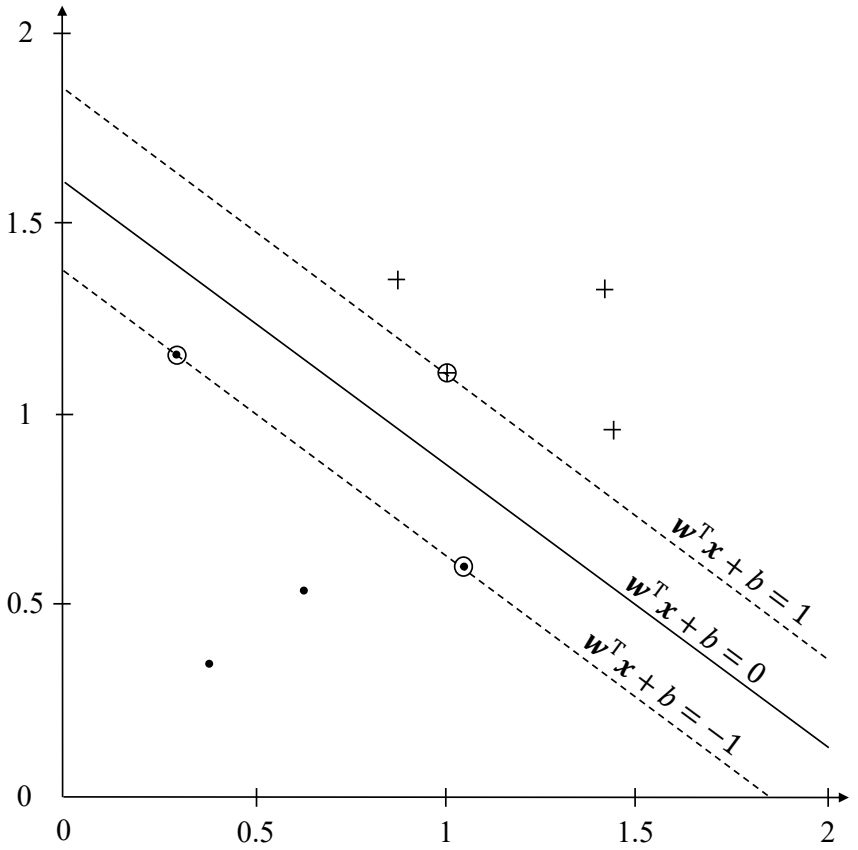


Fig. 4.2 2D example for a SVM (According to [5]).

The general concept is to calculate not only a hyper-plane which just separates the points of the two classes but a hyper-plane for which each point has a certain margin to the plane. For the determination of the hyper-plane, only the so-called support vectors are used. In Figure 4.2, these are the instances marked with a circle which lie on or within the margin. Since the complexity of the determination of the hyper-plane does not depend on the dimensions of the high-dimensional space but on the number of points considered for the calculation of the hyper-plane, the problem can be significantly simplified by using the support vectors [5].

Decision Trees and Random Forests

Another approach for supervised classification are decision trees which learn to classify data instances by a recursive partition of the input space [106]. An example of a decision tree is shown in Figure 4.3 [39]. The decision tree consists of a root node, several internal nodes and leave nodes which are connected by edges from top to bottom. Each node divides the input space into sub-spaces with regard to some input attributes. Each leave node is assigned to one of the classes corresponding to the problem statemen [106]. Figure 4.3 is showing a binary decision tree where the root node as well as all internal nodes have exactly two leaving edges.

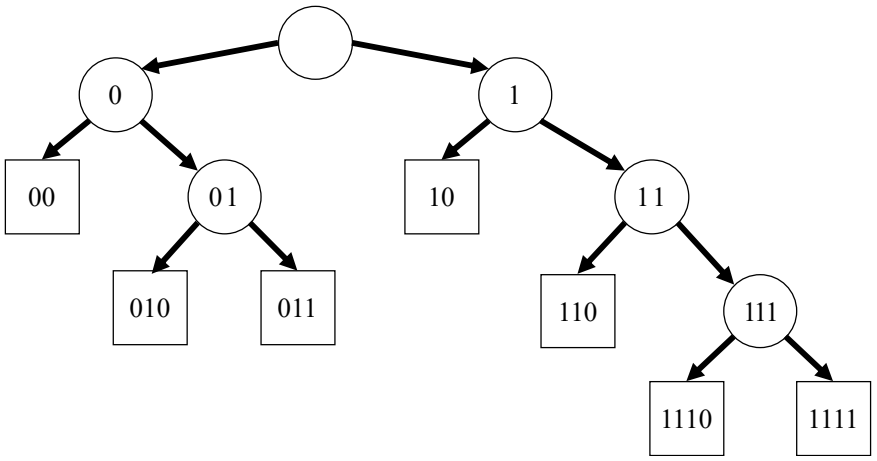


Fig. 4.3 Example of a decision tree [39].

The tree structure is created based on a set of labeled data instances. The data is recursively split based on the most discriminative feature values. In the root node, the whole set is split the first time and subsequently the corresponding sub-sets of each internal node are split again regarding another feature or another threshold value. A leave node is found as soon as all instances of a sub-set belong to the same class [4].

One major issue of decision trees is that they are learning axis-aligned decision boundaries. Therefore, they struggle when the real decision boundaries in the underlying data are not axis-aligned [39]. A big benefit, however, is that their simple structure allows humans to interpret their decisions [4].

Based on decision trees, the method of random forests has been developed. These can be used for both regression and classification and, as the name suggests, are generated from several decision trees. Instead of using only one decision tree, several trees are used and the decision for e.g. a classification task is made by majority voting. The individual trees are each generated on the basis of a randomly selected partial data set of the original data.

Furthermore, only a subset $m << M$ of the M dimensions, i.e. features of the data space, is used as a criterion for a split in a node. This subset of features is again randomly chosen for each tree. Advantages are that the system can be trained and evaluated relatively fast since the different trees can be trained and evaluated in parallel. In addition, random forests often generalize very well and therefore do not tend to overfit [17].

The different ML algorithms described above are appropriate solutions for a wide range of tasks. Nevertheless, they reach their limits when trying to solve complex problems like speech or object recognition. The more complex a problem becomes e.g. when it is extremely high dimensional, the more difficult it becomes for an ML algorithm to solve the problem. In these cases, DL algorithms are often more powerful than traditional ML algorithms [39]. Before we explain the domain of DL algorithms in Section 4.2, we will first discuss three techniques which are of crucial importance for the development of our solutions for the two process steps *Manufacturability Analysis* and *3D Component Recognition*: The method of transfer learning, learning from synthetic data and data augmentation.

4.1.5 Transfer Learning

While working on the two process steps *Manufacturability Analysis* and *3D Component Recognition*, we encountered the problem that although information was available in principle for addressing the issues, it could not be used directly for training data-driven solutions. Therefore, in order to develop and use a suitable ML respectively DL solution, knowledge must be transferred from other domains or tasks and synthetic data must be generated and augmented. In this sub-section and in the following two sub-sections, we will describe concrete methods which we used to address the problem of data scarcity. Without these methods, it would not have been possible to realize the corresponding solutions.

In order to solve a problem with ML methods, corresponding data, usually even a large amount of data, is needed first, on the basis of which the ML model can be trained. The training data should additionally come from the same statistical distribution as the data used later in the application. However, often the problem arises that there is not enough data to solve a problem using an ML model and it would be very expensive to generate it. In this cases, so called *transfer learning* can be used for transferring knowledge from an other domain to the problem domain and/or from an other task to the problem task. As a result, problems can often be solved on the basis of a significantly smaller amount of data from the problem domain itself [97]. In this section, we will give a brief overview about different transfer learning techniques.

Figure 4.4 shows an overview about the three different transfer learning settings *inductive transfer learning*, *unsupervised transfer learning* and *transductive transfer learning* which are defined via the relation of source and target domain D_S and D_T and source and target task T_S and T_T. When the source and target domain, as well as the source and target task are the same, classic ML procedures are used without any need for transfer (here mentioned as *traditional machine learning*) [97].

Learning Settings		Source and Target Domains	Source and Target Tasks
Traditional Machine Learning		the same	the same
Transfer Learning	*Inductive Transfer Learning* / *Unsupervised Transfer Learning*	the same	different but related
		different but related	different but related
	Transductive Transfer Learning	different but related	the same

Fig. 4.4 Overview of the three different transfer learning settings [97].

For the *inductive transfer learning* situation, the source and target domain are the same but the tasks differ. In this case, based on a different source task T_S, the ML model can gain information about the source domain D_S which is equal to the target domain D_T. Subsequently, by induction, this information can be used to train the ML model to solve the target tasks T_T using a relatively small amount of data related to that target task [97].

If the source and target domain are different but the tasks are the same, we speak about *transductive transfer learning*. In this case, the ML model is learning the source task T_S which is similar to T_T in the source domain D_S. The trained ML model can subsequently be adapted to the target domain D_T using a relatively small amount of data from that domain [97].

In the *unsupervised transfer learning* case, the source and target domain as well as the source and target task are different and no labeled data is available for both cases. The ML model learns to solve unsupervised learning tasks in the source domain D_S to enhance its ability for learning a different but similar task using unsupervised learning in the target domain D_T. Depending on the problem, all these approaches offer the possibility to use ML methods also for tasks where only a relatively small amount of training data is available. This significantly expands the application possibilities for ML methods [97].

The described method of transfer learning is a first possibility to extend the application possibilities of ML and åDL solutions and therefore forms a first important building block for the development of our solutions. If, however, no data from the target domain and task is available at all for the training of a solution, which is the case for the process step *3D Component Recognition*, it is necessary to generate synthetic training data. This will be discussed in the following subsection.

4.1.6 Learning from Synthetic Data

As already explained in Section 4.1.5, the training of an ML model usually requires a large amount of data corresponding to the problem domain and task. However, depending on the research question or domain, often no or only a very small amount of data is available because gathering and annotating big data sets is often very expensive and time-consuming [115]. In order to be able to process a problem with ML methods, alternatives to expensive data collection and labeling must be created. One possibility is the generation of synthetic training data which can be used for transductive transfer learning as described in Section 4.1.5. Especially in the field of image processing, several approaches to generate synthetic data which can be used for the training of ML models, have been developed in the last years. Based on virtual scenes, virtual images of e.g. objects to be recognized can be rendered. In order to generate appropriate synthetic data for a specific problem, some guidelines must be followed. We will explain those guidelines on the example of the generation of synthetic images, since this is relevant for our solution described in Chapter 7.

Since the ML model trained on the basis of the synthetic training data is usually used to evaluate real images after training, it is decisive how well the synthetic training data is adapted to the real conditions, in order to achieve the best possible performance. Small changes between training and inference data can lead to enormous variations for the performance of the ML model. To generate optimally adapted synthetic images, the following aspects must be considered [48]:

1. Additional to the 3D objects which shall be recognized, segmented or identified in the inference stage, the whole scene with the objects environment has to be virtually modeled as detailed as possible.

2. A high photorealism is necessary. Besides an accurate modeled virtual geometry of the 3D object, also realistic material and textures are necessary. This can be achieved by photorealistic rendering.

3. The whole scene and its geometric configuration have to fit to the physical situation. That includes the virtual positioning of the camera(s) and light source(s). Different view points and light source positioning lead to different perspectives or shadow casting and therefore influence the performance of the ML model. Especially in our case, the recognition of monochrome objects, the shadows provide a large amount of information and therefore have a high value.

If the points mentioned above are adapted to the specific situation, realistic training data can be generated. This offers two possibilities: On the one hand, ML models can be trained

completely without physically generated training data. On the other hand, the ML models can be pre-trained using the synthetic data and then adapted to the physical problem using a small amount of real data. In both cases, the use of synthetic data can save time and costs for collecting and labeling physical training data and still produce comparable results [52].

4.1.7 Data augmentation

In the previous two sections, we already described two methods which enable ML models to be used also if no or only a small amount of training data from the problem domain is available for a given task. Using transfer learning and synthetic training data, we can in principle already address the issue of *3D Component Recognition*. Since this process links the information of digital 3D models with physical components, it is important to optimally align these two domains. Data augmentation methods can be used to process existing data accordingly and thus improve the quality of the data. In our case, this means that the synthetic training data generated on the basis of the digital 3D models can be adapted to the physical components using augmentation techniques. Training data in the form of images, which are the data representation used by our solution for the *3D Component Recognition*, can e.g. be extended by photometric or geometric transformations to create additional training data instances to enable the training of an ML model and prevent overfitting [22, 123]. In this section, we will give a brief overview about some of the basic photometric and geometric transformations methods for the example of image augmentation. The image augmentation techniques are relevant for the development of our solution of our solution for the *3D Component Recognition*. However, the geometric transformations can also be applied directly to 3D models. We use those methods for our solutions for both problems, the *Manufacturability Analysis* and the *3D Component Recognition*. Besides those techniques, several additional methods for image augmentation exist. Nevertheless, we are focusing on these since they are the most relevant techniques for our specific solutions.

Geometric Transformations

Geometric transformations can be used as a simple method to modify images to create additional training data. Some common methods are the following.

1. Non-distorting operators: Standard operators like rotation, translation, flipping or cropping can be used to create additional images without distorting the actual geometric ratios [123].

2. Distorting operators: Images can be distorted using operators like grid distortion or elastic transformation. These operators locally deform the images and therefore change the geometric ratios of the image [22].

Photometric Transformations

Photometric transformations change properties such as brightness, color, sharpness, or saturation of an image at the pixel level. This is often very helpful, since there are sometimes strong variations regarding those properties when generating real images. The following list provides an overview about common photometric transformation methods [22].

1. Color changes: The color of an image, i.e. the corresponding red, green and blue (RGB) values of the pixels can be changed e.g. by RGB-shifts, hue saturation change, color histogram equalization or shuffling of the RGB channels.

2. Contrast, Gamma or Brightness randomization: By randomly generated changes of these properties, additional images can be generated

3. Blurring: An image can be blurred using local filters like e.g. a median filter.

4. Noise injection: Gaussian noise can be added to the original image.

5. Sharpening: Images can be sharpened e.g. using a high-pass filter.

The methods described above offer the possibility of extending data sets or adapt the training data to a specific situation. In Chapter 8 and 9, we will explain how these methods are used to support the solutions which we designed for the tasks of *Manufacturability Analysis* and *3D Component Recognition*.

4.2 Deep Learning

In Section 4.1, we described the basics about ML and already mentioned that traditional ML algorithms often reach their limits when the problems to be dealt with becomes too complex due to e.g. high dimensionality. Often, daily problems such as speech or facial recognition, which are easy tasks for us humans, are difficult to solve using traditional ML algorithms. At this point, DL models offer large potentials. In the last, decade DNNs have achieved outstanding performance in this type of tasks and have greatly improved the state-of-the-art compared to ML approaches [72].

The two sub-processes *Manufacturability Analysis* for AM and *3D Component Recognition* of additively manufactured components both belong to the group of complex problems. For the *Manufacturability Analysis*, often complex three-dimensional relationships are decisive for the manufacturability of an object. These can depend on combinations of local and global geometric properties of the objects. The solution must therefore be able to process detailed, fine-grained information in such a way that links can be created at different hierarchical levels to determine a manufacturability evaluation for on object. Due to this complexity, traditional ML algorithms are probably no longer suitable for solving the problem. The *3D Component Recognition* problem is a classical recognition task. In the last years, DL algorithms have proved that they are outperforming traditional ML algorithms for this type of tasks [146]. Therefore, the solutions which we developed for the two sub-processes *Manufacturability Analysis* for AM and *3D Component Recognition* are based on DNNs. We will provide the necessary information for the development of the solutions in this section. We will begin with a basic overview of the structure and work flow of DNNs in general and subsequently explain the specific architecture of CNNs which are specialized for the processing of images or acoustic signals.

4.2.1 General Structure

Similar to the structure of traditional ML models, which we explained in Section 4.1.3, the creation of a DL model includes the same four building blocks: a data set, a DL model, a cost function and an optimization algorithm. We will go through these steps using the example of an Multilayer Perceptron (MLP) which is a standard Feedforward Neural Network (FNN). We will skip the description of the data set at first since it is not necessary for the explanation of the basic structure of a DL algorithm.

Network Architecture

The basic structure of an MLP is shown in Figure 4.5 [40]. The term deep in DNN is characterized by the concatenation of many layers of the DNN. This creates the depth of the DNN. This allows DNNs to represent significantly more complex mathematical constructs than classical ML approaches. MLPs and also all other DNN types consist of several layers which again consist of several nodes, so called neurons. The neurons from adjacent layers are linked by weighted connections. The example shown in Figure 4.5 is showing so-called fully connected (FC) layers. That means that all neurons of a layer l are connected with all neurons of the previous layer $l-1$. In the case of FNNs, these connection are exclusively forward directed. As can be seen in Figure 4.5, these layers are divided into an input layer, one ore

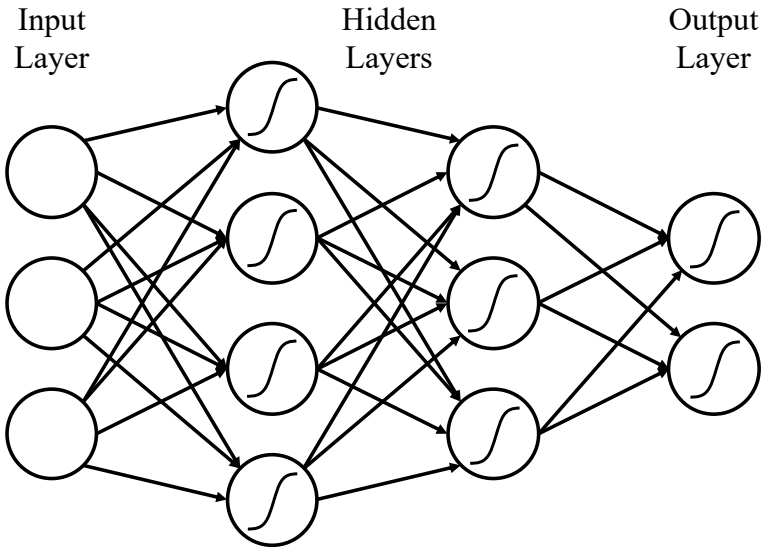


Fig. 4.5 Structure on an MLP network, a classic FNN (Adapted from [40]).

more hidden layers and an output layer. Based on input data, the neurons of the input layer are activated and the corresponding activations are propagated through the hidden layers to the output layer and the corresponding neurons. Each neuron represents a computational cell which is defined by an activation function. While the input units are normally only representing the input data, the neurons of the hidden layers and the output layer are generally defined by a non-linear activation function like the logistic sigmoid function [40].

Similar to traditional ML techniques, DL models have to be trained prior to their application. The training step consists of two main steps, which are iteratively repeated. In the forward step the output for e.g. a classification task is calculated. Subsequently, the backpropagation step is executed. In this step, the error which is defined by an error metric is propagated back to the input for being able to adjust the weights of the connections between the neurons of the network and thus optimize the whole model. We will explain the two steps in more detail [72].

Forward Step

In the forward step, an input instance is processed and an output decision, e.g. a class for classification tasks, is calculated by propagating the activations through the network. For this purpose, an activation function is selected for each layer respectively the corresponding neurons. This can e.g. be the hyperbolic tangent or the logistic sigmoid function. However, depending on the problem, the activation function can be adapted [40].

The I input neurons directly contain the data of an input data instance. These are passed to the first hidden layer using a weighted sum of the inputs with the weights w_{ih} between an

input neuron i and hidden neuron h. The activation a_h of a neuron h of the first hidden layer is shown in Equation 4.6 [40].

$$a_h = \sum_{i=1}^{I} w_{ih} x_i \tag{4.6}$$

The final activation b_h of the neuron h of the hidden layer is subsequently calculated using the activation function (See Equation 4.7). As activation function Θ_h, non-linear functions like the hyperbolic tangent or the logistic sigmoid function are commonly used. The final activation b_h is subsequently propagated to the next hidden layer the same way [40].

$$b_h = \Theta_h(a_h) \tag{4.7}$$

This propagation from layer to layer continues until the activation b_h of the last hidden layer reaches the output layer. The activation a_o of an output neuron is again calculated via a weighted summation over the activations of the neurons of the last hidden layer. The activation function of the neurons of the output layer is strongly related to the respective task. For binary classification tasks, the logistic sigmoid is the standard function, for multi-class classification tasks the softmax function is used. For regression problems, often no output activation function is used, since the actual numerical values calculated by the network are of interest [40].

Using the described functionality, the network calculates the output regarding a certain issue based on an input data instance. For being able to train the network, an error metric is needed.

Error Metric

The error function also depends on the problem being processed. For regression problems, often the Summed Squared Error (SSE) metric is used (See Equation 4.8). The error E is calculated by summing up the squared distance between the calculated output y_o of an output neuron and the corresponding ground-truth target t_o^n over all N input data instances and all O output neurons of the network [116].

$$E = \frac{1}{2} \sum_{n=1}^{N} \sum_{o=1}^{O} (y_o(\mathbf{x}_n; \mathbf{w}) - t_o^n)^2 \tag{4.8}$$

For classification tasks, the cross-entropy function is most common. The corresponding equation is shown in Equation 4.9. For classification problems the amount of output neurons is usually equal to the number of classes. Sometimes, depending on the problem which is

dealt with, an additional unknown class is added. The error E is calculated by summing up the multiplication of the target t_n^o corresponding to an input data instance and the ln of the calculated output over all N input data instances and all output neurons O. The target t_n^o is equal to 1 if o is the correct class and 0 if not [116].

$$E = -\sum_{n=1}^{N}\sum_{o=1}^{O} t_o^n ln(y_o(\mathbf{x}_n;\mathbf{w})) \tag{4.9}$$

The error metrics are optimized during the training step. This is done using gradient descent in the backpropagation step.

Backpropagation

In the backpropagation step, the weights of the connections between the neurons are adjusted to minimize the chosen error metric. For the optimization, the derivative of the error function with respect to the weights is used and the weights are changed in the direction of the negative gradient (See Equation 4.10) using a step size η. The error and its gradient are propagated back layer by layer from the output to the input of the network by using the chain rule for partial derivatives [40].

$$\Delta\mathbf{w} = -\eta\frac{\delta E}{\delta \mathbf{w}} \tag{4.10}$$

Using the backpropagation algorithm, a NN can be optimized by iteratively applying the forward step and subsequently the backpropagation step using the training data.

4.2.2 Convolutional Neural Networks

In Section 4.2.1, we explained the general architecture and working principle of DNNs. Based on this architecture and working principle, various network architectures have been developed which are optimized for solving specific classes of tasks. The two most important network architectures are Recurrent Neural Networks (RNNs) [41] which are optimized for processing sequential data like speech signals or texts and CNNs [71] which are optimized for processing of grid-structured data like images which are represented by 2D grids of pixels or 3D models, which can be represented as 3D grids of so called voxels.

For the development of our solutions for the two process steps *Manufacturability Analysis* for AM and *3D Component Recognition* of additively manufactured components, we use both image-based and 3D data-based CNN architectures. Therefore, we will explain their working principle in more detail in this section. The success of CNNs is based on four key

features of a CNN: *locality*, *weight sharing*, *pooling* and the *depth* of the network [72]. In the next sub-sections, we will deal with the individual points.

Convolution

The *locality* is provided by the central element of a CNN, the convolution operator. Although, in principle the operator can be applied to data of arbitrary dimensions, we explain it using 2D data as an example. The convolution operator is used to calculate a so called feature map S of the input data I by using a convolution kernel K. The input data and the convolution kernel are represented by two dimensional grids. The calculation of the feature map is shown in Equation 4.11 with a convolutional kernel K of size $m \times n$ [39].

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{4.11}$$

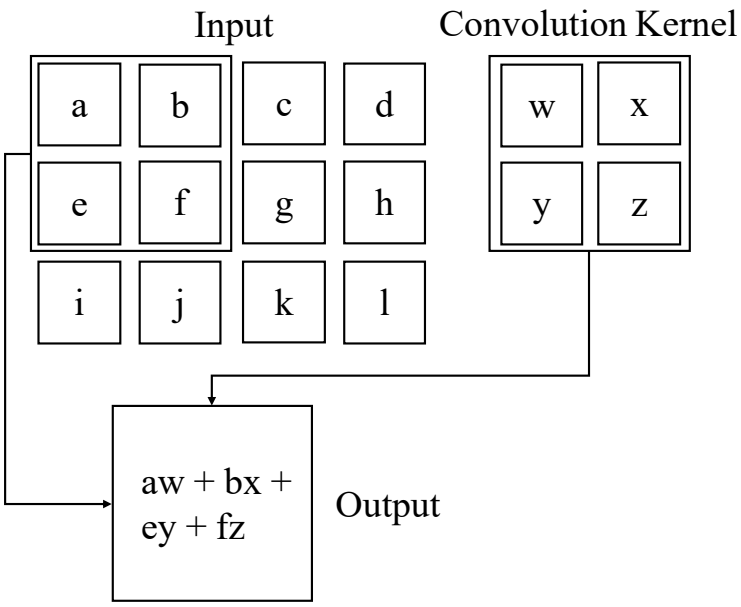


Fig. 4.6 2D example of the convolution operator (Adapted from [39]).

For illustration, a 2D example of the functionality of the convolution operator is shown in Figure 4.6. It shows the example calculation for the feature map $S(1,1)$, assuming that the indices start at 0. The feature map is calculated by multiplying the values of the input data with the corresponding value of the convolution kernel [39].

Based on the convolution operator, convolutional layers are designed. Compared to the FC layers described in Section 4.2.1, there are several different properties. The first is that convolutional layers generate a sparse interaction with the neurons of the previous layer.

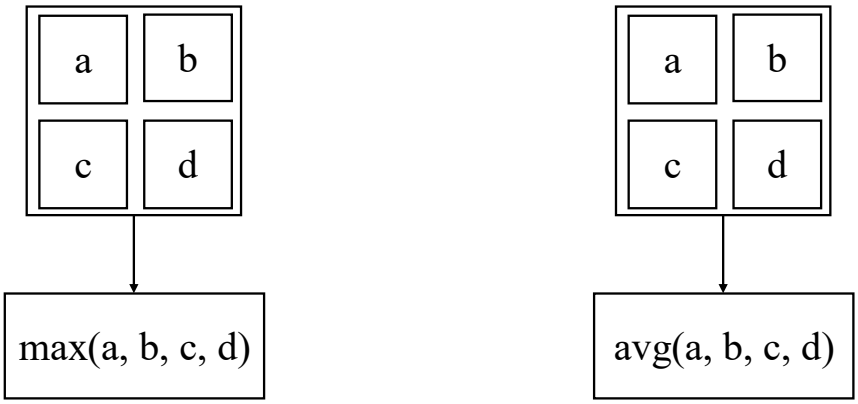


Fig. 4.7 Structure of max pooling(left) and average pooling (right) in 2D.

Instead of being connected with all neurons of the previous layer, the neurons i.e. the feature maps of a convolutional layer are only connected to the neurons of the previous layer in the local neighborhood inside the kernels dimensions. Thus, the convolution kernel focuses on a local area and is able to learn local image features like edges [39].

A second advantage which we already mentioned at the beginning of this section, is the *weight sharing* of a convolution kernel. Instead of learning each weighted connection between the neurons of adjacent layers separated like it is done for FC layers, a certain amount of different convolution kernels is learned during the training. Each convolution kernel has a size $m \times n$ and accordingly $m \times n$ weights that can be learned during training. Each kernel is then over the whole grid of input data respectively grid of feature maps of the previous layers. This allows kernels to learn to recognize properties such as an edge in a certain orientation in general. By sliding the kernel over the grid, this type of edge can be detected anywhere in the image, regardless of its position. This means that the learned kernels are equivariant in terms of translation [39].

The convolution operation is one of the main elements of CNNs. However, the high performance for solving problems such as object recognition is only possible with a suitable network architecture consisting of further layer types.

Pooling

The *pooling* operator is the second central element of a CNN. It is used to summarize the information in a local area. Two examples are shown in Figure 4.7. The max pooling operation is forwarding the maximum value within the receptive field of the pooling kernel which has a size of 2×2 in the shown example. The average pooling kernel is forwarding the average of the values inside the receptive field of the kernel. Pooling is used to make the feature representation nearly invariant to small translations in the input data. This enables

CNNs to recognize decisive structures in the input data with more robustness regarding translations in the data. Additionally, they are reducing the dimension of the features what leads to an hierarchical structure [39].

Network Architecture

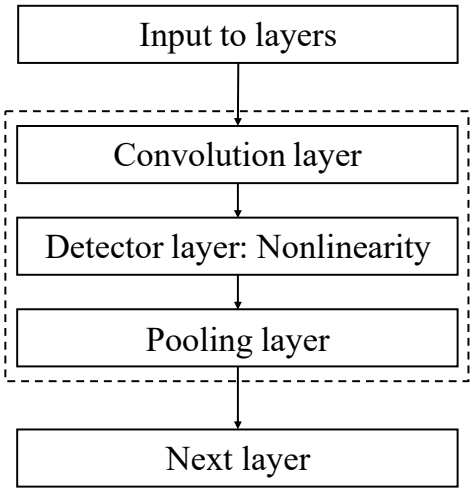


Fig. 4.8 The main building block of a CNN: Combination of convolutional layer, non-linearity and a pooling layer (Adapted from [39]).

An important aspect for achieving the state-of-the-art performance for several tasks like object recognition is the *deep architecture* of CNNs. In most architectures, standard building blocks are used which are stacked in several levels. The standard building block is shown in Figure 4.8. It consists of a convolutional layer, a non-linearity like e.g. a Rectified Linear Unit (ReLU) and a pooling layer. The convolution layer is learning multiple convolutional kernels in parallel what enables it to learn several different features like e.g. edges in images. The whole CNN often consists of various of theses blocks followed by FC layers and a specific output stage chosen with regard to the corresponding task to be dealt with [39].

This deep structure makes CNNs extremely powerful. While the first layer is able to learn local features, the hierarchical structure enables the deeper layers to learn global features which learn the properties of the input data as a global context based on the low level features of the first layers.

This structure has made CNNs extremely successful for handling various tasks such as the ModelNet40 object recognition challenge where CNN-based approaches outperform traditional object recognition algorithms [146]. For this reason, CNNs also have a decisive importance for the development of our own solutions for processing the two process steps

Manufacturability Analysis and *3D Component Recognition*. In the following section, we will briefly explain transfer learning techniques for DL and subsequently explain some state-of-the-art CNN architectures from the domain of 3D DL which are designed for solving tasks related to the process steps of *Manufacturability Analysis* and *3D Component Recognition* and therefore form the basis of possible solutions for our application.

4.2.3 Transfer Learning for Deep Learning

In Section 4.1.5, we explained the method of transfer learning for ML algorithms. Since we are using DL methods for the development of our solutions, we address the concrete implementation of transfer learning for DL in this section. Transfer learning is also a very effective technique for training DNNs. Due to the deep hierarchical structure of DNNs, there are several possibilities to use transfer learning for DNNs. In the following, we refer to CNNs because those are the most relevant type of DNNs in the context of this thesis.

The deep hierarchical structure of CNNs causes the lower layers to learn local geometric properties, which are used by the deeper layers of the CNN to generate high level features. Therefore the low level features generated by the lower layers are relatively universal and problem independent. In contrast to that, the higher layers are much more problem dependent. This fact can be used to apply different types of transfer learning for CNNs, depending on the actual problem setting and the corresponding data set [81].

In general, often *transductive transfer learning* as explained in Section 4.1.5 is used for transfer learning for CNNs. That means that the source task T_S is often similar to the target task T_T, the source domain D_S can be both, different or similar to the target domain D_T. Similar to the general transfer learning techniques explained in Section 4.1.5, the choice of the transfer learning variant is dependent on the size of the target data set and the similarity of the target domain D_T and the source domain D_S [121]. We will first explain the general possibilities for different transfer learning variants for DNNs and subsequently explain which variant is used in which case.

As already explained in Section 4.2.2, a CNN can be split into two parts. The first part is the feature extraction stage which is composed of multiple iterations of convolution blocks. The second part is the classification stage for classification problems respectively a regression stage for regression problems. Depending on how the source and target domains D_S and D_T are related and how much data is available in the target domain, there are different options which parts of the CNN remain frozen after the pre-training and which are adapted to the target domain D_T by fine-tuning. The classification/regression stage is the most problem- and domain-dependent part of the CNN. Therefore, it is always either fine-tuned or completely

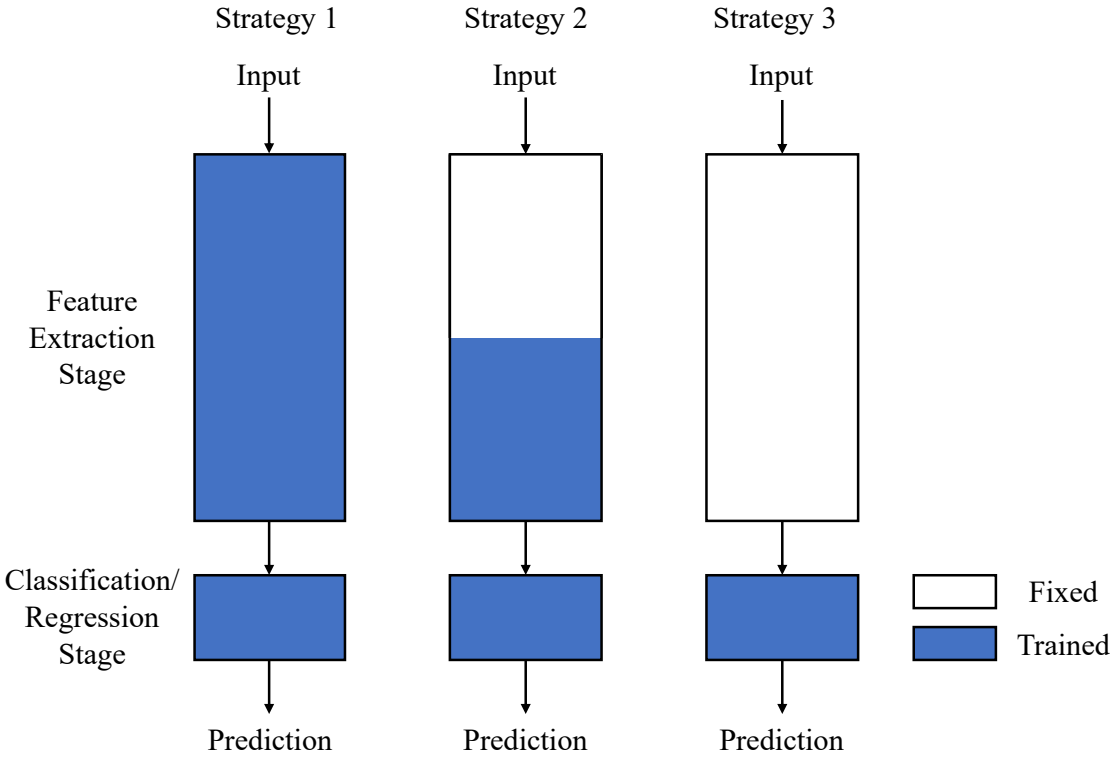


Fig. 4.9 The three basic variants of transfer learning for CNN (Adapted from [81]).

trained from scratch using the data set of the target domain. The feature extraction stage can either stay fixed or be partly or fully fine-tuned using the data from the target domain [81].

Figure 4.10 shows how the choice of fixed parts of the CNN and parts to be fine-tuned can be chosen based on the size of the target data set and its similarity to the source data set. When the target data set is different from the source data set, it is often necessary to fine-tune larger parts of the CNN. If enough data is available in the target domain, the complete CNN should be fine-tuned based on the pre-trained CNN to achieve the transfer to the target domain. If the target data set is smaller, at least parts of the pre-trained CNN should stay fixed to prevent overfitting. On the other hand, it is necessary to fine-tune parts of the CNN for being able to transfer between the different domains of the source and target domain. For the case of a large target data set and similar domains, a smaller part of the feature extraction stage has to be fined tuned, since the lower layers already learned low level features based on data from a very similar domain. When having only a small data set from the target domain but a similar source and target domain, only the classification/regression stage should be fine-tuned respectively retrained to prevent overfitting [81].

The application of transfer learning offers various possibilities. On the one hand, DNNs can be used for solving problems where only a small amount of data is available. On the other hand, the use of a pre-trained DNNs can often both, reduce the training time and improve the

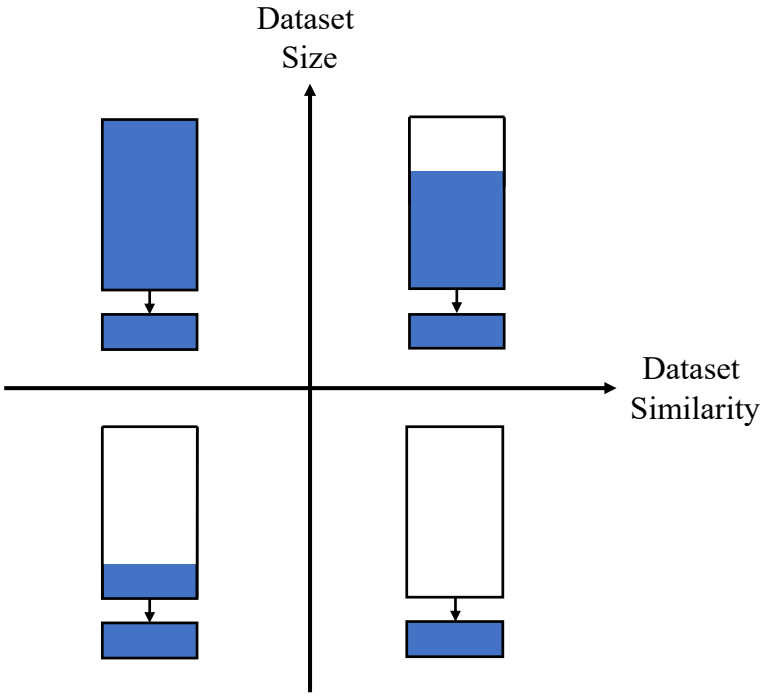


Fig. 4.10 Transfer Learning variant in relation to the size of the target data set and its similarity to the source data set (Adapted from [81]).

performance with regard to the specific problem [50, 96]. These characteristics make transfer learning especially valuable for our solution concept for the process step of *3D Component Recognition* for additively manufactured components.

The described approaches form the basis for a variety of complex ML and DL architectures that are used for CV tasks which are similar to the process steps of *Manufacturability Analysis* and *3D Component Recognition* in the AM domain which we are dealing with. We will explain these approaches and also traditional CV approaches in Section 4.4. First, however, we will deal with the topic of XAI which is relevant for the development of a solution for the *Manufacturability Analysis* for additively manufactured components.

4.3 Explainable Artificial Intelligence

Parallel to the increasing interest of research and industry in the further development of ML and DL algorithms, the interest in the explainability of the algorithms is simultaneously growing. Understanding the decision of an ML model offers several advantages. For example, the explanation can help to better understand the ML model and thus be able to optimize the

model. Additionally, the explanation can even be an essential part of the application itself [10]. Therefore, the focus of the research community on this topic is expanding rapidly.

As stated in Section 3.1.3, besides the binary decision if an object is manufacturable or not, a suitable solution for the *Manufacturability Analysis* has to be able to provide feedback with regard to critical sub-geometries of a 3D model. For this purpose, we are using methods from the field of XAI for DL. In this section, we will therefore first explain the necessity of XAI and subsequently give an overview about basic approaches for explainability for ML and DL.

4.3.1 Necessity of Explainability

In the last years, ML and DL algorithms entered more and more fields of application and outperformed traditional algorithms. Meanwhile, they represent the state-of-the-art in fields of application such as image, text or speech processing [117]. With growing complexity of applications, also the complexity of the ML models is growing. The decisions of simpler ML models such as decision trees can be reproduced without much effort due to their significantly lower complexity. This is called a *transparent box design*. With the increasing complexity of ML models, their performance potential increases which however is accompanied by a loss of explainability. The inner logic of these models is no longer understandable for a user. Therefore, these ML models are called *black boxes* [43]. However, for a variety of reasons, explainability is essential for both researchers and users of the ML models [45].

The main reasons why explainability is necessary are shown in Figure 4.11 from the perspective of different stakeholders.

The first party is the group of ML developers. With the help of explanations regarding the decision making of the ML models, developers are able to better understand the models. Using that knowledge, developers can be able to further optimize the models.

In addition to the developers, explainability is also an important aspect for many users of ML models. One common example for the necessity of explainability for an application is the usage of ML for medical tasks [136]. Nowadays, especially DL models are widely used for the interpretation of medical images, achieving results at the level of medical experts. Therefore, DL models are often used to support the experts in their medical decisions. Since the health or even life of a patient can depend on the interpretation of medical images, a correct interpretation is extremely important. In order to ensure that medical personnel can rely on the results of the DL models, the models must be able to explain their decisions [66].

Another aspect relates to the fairness of the models. ML models are entering more and more into daily applications. For example, they are used for credit ratings or for the analysis of application documents. For these applications, there is a potential risk of unfairness and

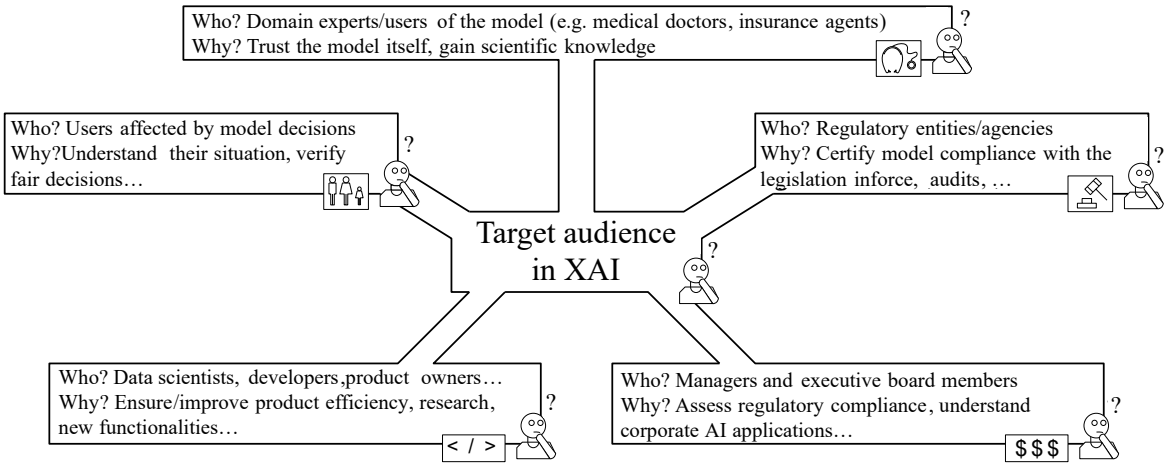


Fig. 4.11 Different reasons for explainability [10]

discrimination regarding characteristics like race, gender or religion [18]. Explainable ML models allow such cases to be identified and the models to be adapted.

These are just some of the reasons that show that with increasing complexity and performance of ML models and the associated broader use in real-world applications, the need to explain the models is growing, too. Therefore, XAI is currently a highly focused field of research. In the following section, we give an overview of explainability methods for ML and especially DL models.

4.3.2 Methods for Explainability

There are several basic approaches to explain ML models. These are shown in Figure 4.12. One option is directly designing the ML model as a *transparent box*. These models are designed to enable the user to directly understand the reasons for a decision. For complex ML models like DNNs this design is not possible. For these *black boxes*, the explanation can be divided into *Model Explanation*, *Outcome Explanation* and *Model Inspection* [43].

Model Explanation tries to capture the whole logic of a black box model and create a new ML model which shows the same behaviour and is transparent. In contrast, *Outcome Explanation* does not attempt to understand the entire logic of a black box, but only to determine the explanation for individual input data instances. *Model Inspection* is focusing on explaining parts of the black box by e.g. changing an input attribute and observe the sensitivity to these changes [43].

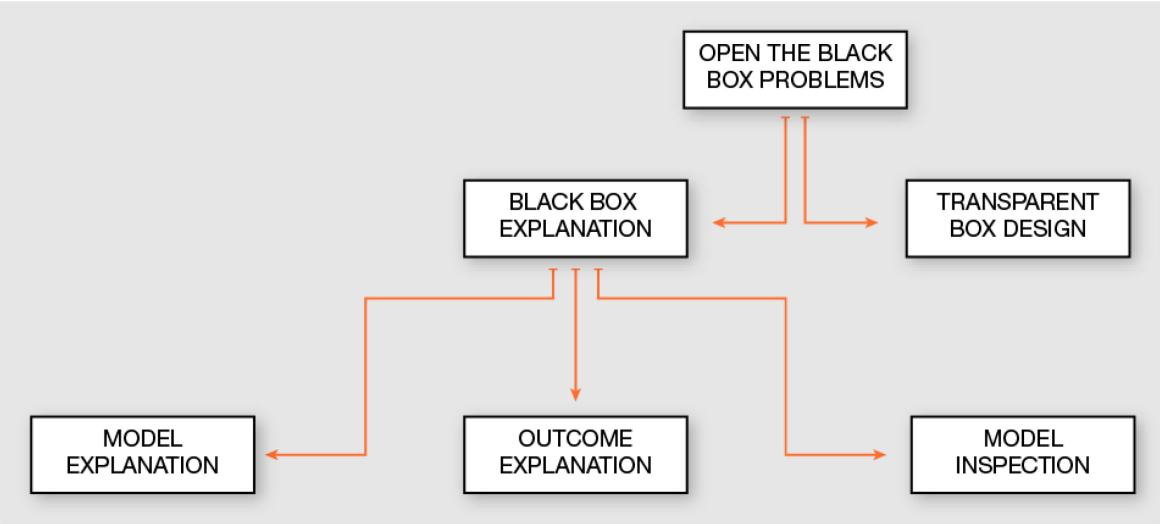


Fig. 4.12 Different approaches for black box explanation [43]

4.3.3 Explainability of Deep Neural Networks

Approaches for explainability for DNNs can be divided into similar groups as the general groups explained in Section 4.3.2. The category of *Model Explanation* approaches is dealing with transparent box models which are copying the behaviour of the DNN and are additionally able to provide an explanation for the behaviour. For generating an *Outcome Explanation* mainly visualization methods are used which highlight decisive characteristics in the input data. In *intrinsic methods*, the DNNs are designed such that they not only generate the actual output, but also an explanation for it [147]. These approaches try to enable an at least partially transparent behavior directly on the design level through their architecture.

For giving an overview, we will briefly explain the different groups of methods and subsequently go into detail for the method used in this thesis.

Model Explanation

The general procedure of *Model Explanation* is shown in Figure 4.13. A new interpretable ML model g is trained to approximate the behaviour of the original DNN f so that $g(\mathbf{y}) \approx f(\mathbf{x})$. $\mathbf{x}$ is the original input data set and $\mathbf{y}$ contains both the original input data and the features learned by the DNN. Therefore, the new model g has access to additional information what enables it to learn the behaviour of the original model f while being simpler and interpretable [147].

By using the additional information which already have been learned from the DNN, the simpler ML model g may be able to achieve a comprehensive performance for the task at

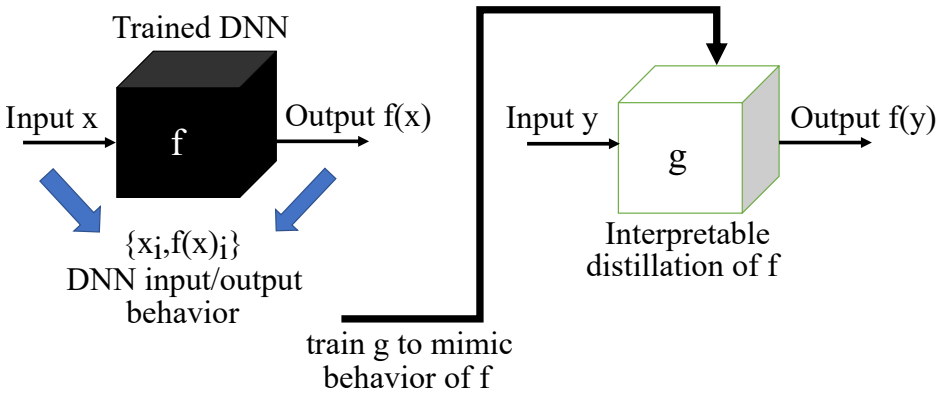


Fig. 4.13 An interpretable model g is trained to approximate the behavior of the original model f [147].

hand, even though it is simpler than the original model f. Through the connection of the original input data **x** and the output **y** of the DNN, the new model g can generate information about the way the input data **x** affects the decisions of the DNN. Although this information can not explain the complete behavior of the DNN, it does provide at least some insights into the behaviour of the DNN and makes it more explainable [147].

Outcome Explanation

Outcome Explanation is a currently widely regarded field of research to explain the behaviour of DNNs. Especially when using CNNs, *Outcome Explanation* methods are used to visualize the reasoning behind the decision of the Neural Network (NN). There are several groups of approaches which are developed for calculating so called Saliency Masks (SMs) which visualize the decisive areas in an input data instance. This can e.g. be salient pixels in an input image [43].

For calculating these SMs, mainly two common procedures are used: Class Activation Mapping (CAM)-based methods and backpropagation-based methods [55]. A CAM visualizes the regions in an input image which have the highest discriminative information for the decision of a CNN, e.g. for a classification task, by performing Global Average Pooling (GAP) on the convolutional feature maps of the CNN [151]. For this procedure, the fully-connected layers prior to the final layer have to be replaced by GAP. In order to avoid this necessity of changing the network architecture, the Gradient-weighted Class Activation Mapping (Grad-CAM) approach was developed [118]. The explanation of the decision for a class A is calculated by first backpropagating the gradients of the output of class A to the last convolutional layer and then performing GAP on the gradients of the convolutional layers of the CNN afterwards [118]. For that procedure, no architecture change is necessary.

Compared to that, backpropagation-based methods are calculating the relevance of input data for a decision by propagating the contribution to the decision of a CNN backwards from the output layer to the input layer [55]. With that procedure a pixel-wise explanation of the decision can be created. One of the most common approaches is the LRP approach [12]. A big advantage of the LRP approach is that the approach is based on a set o different rules which can be used for the propagation of the relevance. This enables users to create a combination of rules which is optimized for a certain problem and the corresponding network architecture which is used as solution. Thus, it is possible to create visual feedback which is optimized for the specific needs of the user. Since LRP is an important part of our solution concept for the *Manufacturability Analysis* for AM, we will take a deeper look on the LRP algorithm in Section 6.3.1.

Intrinsic Methods

The two methods *model explanation* and *outcome explanation* described above try to explain the behaviour of DNN using an additional ML model or by performing additional calculations subsequent to the calculations which have been performed for solving the actual task. In contrast to that, for the group of the *intrinsic methods*, the architecture of the ML models is directly designed in such a way that alongside learning the actual task, they also learn to generate a corresponding explanation for their decision. Approaches belonging to the class of intrinsic methods can be divided into the two groups of *attention mechanisms* and *joint training* [147].

Attention mechanisms are generating explainable outputs by calculating the attention weights with regard to the input data, i. e. it is calculated where the DNN focuses on. This can e.g. be achieved by adding layers which generate the attention weights. These methods can even increase the performance of the DNN for a specific task [13].

Approaches from the category of *joint training* add an additional explanation task to the actual issue that has to be solved. This task is learned alongside the main task. One option is to generate an explanation in text form. The system could e.g. learn to chose one of multiple possible explanation words or sentences which help the user to understand the decision. A problem of this type of approaches is that the corresponding explanation data is necessary for the training process. Data sets including explanations are relatively rare [149].

In the context of our work, we use methods of XAI to show that our solution for the *Manufacturability Analysis* for AM learns the correct features and, on the other hand, visual feedback can be generated, which is part of the solution requirement itself. In Chapter 6, we will go into more detail about the method which we are using.

4.4 Computer Vision

For both sub-processes, the *Manufacturability Analysis* for AM and the *3D Component Recognition* for additively manufactured components methods from the field of CV have a high relevance. The sub-process of *3D Component Recognition* is a classical CV task itself and a solution for it can therefore be based on existing CV solutions. In contrast, the process step *Manufacturability Analysis* itself is not a classical CV application. Nevertheless, the field of CV includes many methods for the analysis of 3D data. Even though these are usually used for applications differing from geometric analysis of 3D models, they offer many characteristics that we can use as an inspiration for developing a solution for the *Manufacturability Analysis*. Therefore, in this section, we want to give an overview of traditional and ML-based methods from the field of CV. In principle, traditional CV methods are also suitable as possible solutions for the process step of *3D Component Recognition*. Therefore, these will also be briefly explained. Subsequently, we will outline, why ML-based methods are better suited for our needs.

4.4.1 Traditional Computer Vision

In this section, we want to provide a brief overview about traditional CV methods especially in the context of object recognition. Since the described approaches will not be used in the further process of the work, we will not go into details, but describe the basic approach of traditional algorithms. Nevertheless, the described information are essential since they form a part of the basis for the decision making process for our solution concept.

Traditional CV algorithms are mostly using a two-stage approach for the recognition of objects in e.g. images or point clouds. At first a representation format for the actual recognition process is defined. As data representation e.g. pixel data can be used directly or different types of features can be extracted from the base data. The data representation serves the purpose of defining a meaningful database based on which the discriminating characteristics of an object can be described and retrieved. The second step is the matching step. In this step, based on the selected data representation, the CV system tries to match known object representations to find the corresponding objects in the data [99]. We will present a brief overview about data representations, matching algorithms and their advantages and disadvantages.

Feature-based

Many approaches are using different kinds of feature representations. Some of the most commonly used feature descriptors are the Scale Invariant Feature Transform (SIFT) [78], Speeded Up Robust Features (SURF) [16] or Oriented Fast and Rotated BRIEF (ORB) [108] descriptors. Most important for the creation of a feature descriptor is that it is invariant to noise and geometrical or photometric transformations. The procedure for object recognition using feature-based algorithms is divided into three steps. The first two steps are creating the data representation which is necessary for the matching process. First, key points like edges or corners are detected inside the base data. These key points and their neighborhood are subsequently described by the feature descriptors. The final step is the matching process. The feature descriptor representation of the object to be searched is compared with the feature representation of the input data in which the object is searched [16].

1. Key point detection: For the detection of key points, several different algorithms exist [47, 85, 107]. The most important property of a key point detector is that it is able to extract the key points independent of e.g. rotation, scaling or shifting of the input data. The results must be repeatable. These properties allow objects to be described and detected based on the key points [107].

2. Feature extraction: The feature descriptors are designed to describe the neighborhood of a key point. The different descriptors consider for example the gradients of pixel values of images and the orientation of the gradients. Similar as the key point detectors, the feature descriptors must be largely invariant with respect to geometric and photometric changes [78].

3. Matching: For the matching process, at first the key points and the corresponding feature descriptors have to be extracted from e.g. reference images of the object to be recognized. Subsequently, these feature descriptors can be searched in new images. This is done using search algorithms which are identifying the most similar sets of feature descriptors in the new images [78].

With this method, based on reference data of the objects to be found, in principle any objects can be found in data to be examined. A major issue of this procedure is that it is relatively computationally intensive.

Shape-based

An other group of algorithms is using shape-based matching [137, 138]. These approaches are again divided into two steps. At first geometric elements such as the edges of an object

or the silhouettes are extracted. Subsequently, the matching is performed based on this representation. Independent of the exact shape representation which is used, the following steps are necessary.

1. Shape extraction: The shape extraction can e.g. be performed via segmentation algorithms. The input data is split into a set of segments. With that procedure the edges between different areas or the whole silhouette of objects can be extracted [138].

2. Object representation: For being able to recognize an object in the input data, again a representation of the object in the matching format is necessary. That can e.g. be an edge model or a silhouette model. This representation can be created by generating different views of a 3D model of the object to be recognized. By rendering an object from the different views, a model view graph can be created that contains the different edge models or silhouettes. The graph is subsequently used for the matching process [137].

3. Matching: During the matching process, the shape representation extracted from the input data is aligned with the model view graph of the searched object. If a correspondingly high match is found, the object is considered as recognized [137].

Similar to the method of feature-based approaches, the shape-based approaches are basically able to find arbitrary objects based on reference data. However, the algorithms have the disadvantage that a model view graph with a large number of different views must be generated for robust detection. Similar to the feature-based approaches, this leads to very long computation times during matching.

Regardless of the data representation used, the general work flow is the same for the different approaches. The object which should be recognized first has to be represented by e.g. a feature- or shape-based descriptor and can subsequently be searched within the input data. The data representation has to be chosen by an expert a priori. The expert has to decide which is the best data representation for the existing problem based on his own experience and information regarding the advantages and disadvantages of the individual approaches. This means that there is no automated data-driven adjustment. This is one of the main arguments why ML-based methods are better suited for the *3D Component Recognition* of additively manufactured parts. For this application it is necessary that the solution system automatically adapts to the daily varying components. This is not possible with traditional CV methods in an automated way. Therefore, in the following section we will discuss the more suitable ML-based approaches.

4.4.2 Machine Learning-based Computer Vision

Until 2012, traditional algorithms, as described in Section 4.4.1, were the state-of-the-art in CV tasks like image and 3D object recognition. In 2011, the approach developed by Sánchez and Perronnin [114] achieved a recognition rate of 74.2 % in the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) challenge which is used for the comparison of image recognition algorithms [111]. In 2012, for the first time a CNN-based approach won the challenge and improved the recognition rate to 83.6 % [63]. Since that year, there have been enormous developments in the field of CV tasks like image recognition and 3D object recognition using CNN architectures.

In this section, we will give an overview about the most important approaches. Since 2015, almost all new approaches in the field of 3D object recognition have been compared using the Princeton ModelNet classification challenge [146]. Since these approaches are the basis for the development of a solution concept for the *3D Component Recognition*, we will discuss the most promising approaches here. The approaches can be divided into groups based on the data structure they are using.

For tasks like 3D object recognition, segmentation or retrieval, DNN approaches using different types of data representations have been developed in the last years which can be divided into the following five classes: Feature descriptor-based approaches are extracting features from the raw input data first to subsequently use these as input data for the DNNs. The second and third group of approaches are directly using 3D data in the form of Red, Green, Blue and Depth (RGB-D) data or point-clouds generated from depth sensors or voxel-grids which represent the 3D data using a 3D grid. Instead of directly using the 3D data, the fourth group is using 2D projections like images of the 3D objects and the last group is using data from hyperspectral cameras [54].

The different data representations have several advantages and disadvantages which we will explain here. Since 3D data-based approaches using point clouds or voxels and projection-based approaches have achieved the best performances for the Princeton ModelNet classification challenge in the last years, we will focus on these three groups of approaches [146].

Point-cloud-based

The main difference of the 3D data types point clouds and voxels is that point clouds represent the 3D data in an unstructured way (See Figure 4.14) while voxel grids are structured. That leads to different types of operators used for the processing of the data in the DNNs. Based on PointNet [102], one of the first point-cloud-based DNN approaches, several different

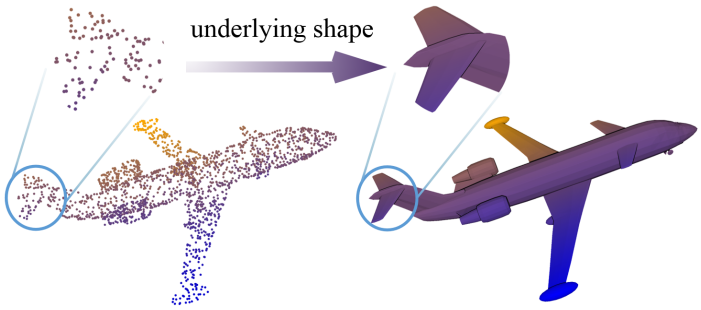


Fig. 4.14 Example of an unstructured point cloud [76].

architectures have been developed in the last years. Pure point-cloud-based approaches can not use the classical convolution operation like it is used by CNNs for image processing since the density of the unstructured point clouds may vary in different parts of the cloud. Instead of the classical convolution operator, local features are learned by e.g. partitioning the point cloud into local sub-clouds and subsequently learn higher level features using a hierarchical structure [104].

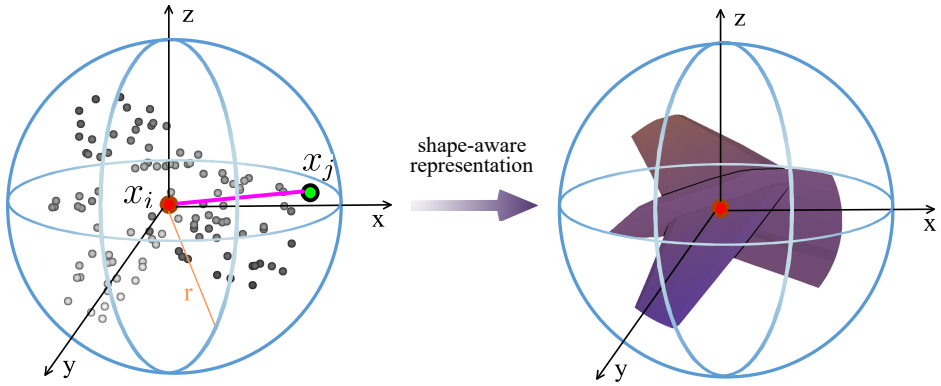


Fig. 4.15 Relation-Shape convolution in a sphere with a radius r (Adapted from [76]).

Currently, the Relation-Shape Convolutional Neural Network (RS-CNN) [76] approach represents the state of the art for point-cloud-based approaches on the Princeton ModelNet classification task achieving an accuracy of 93.6% correctly classified objects for the ModelNet40 classification task [146]. The key concept of their approach is the relation-shape convolution which is able to describe local shapes based on a convolution like operator executed for all points $x_j \in N(x_i)$ within a sphere with a radius r around a point x_i, where $N(x_i)$ are all neighbors of x_i within the radius r (See Figure 4.15). Based on the relation-shape convolution, a hierarchical CNN architecture is designed which is able to learn shape characteristics on unstructured data [76].

A big advantage is that point-cloud-based approaches can directly process the point clouds generated by e.g. a 3D-scanner. The point clouds only have to be sub-sampled to a fixed amount of points for each point cloud in a data set. A limiting factor, however, is that the number of points and thus the resolution of point-cloud-based approaches is limited by the memory of the Graphics Processing Units (GPUs) which are used. Often, only a few thousand points can be used for each point cloud [104, 76].

Voxel-based

In contrast to point-clouds, voxel grids are a structured data representation. Therefore, the convolution operator used by CNNs can directly be used by these types of 3D CNNs. The voxel grids, also called occupancy grids, can be generated from point-clouds or 3D models (See Figure 4.16). Similar to point-cloud-based approaches, a major problem of voxel-based approaches is that the three dimensional data leads to a high memory consumption in the 3D CNNs training process when dense voxel grids with a high resolution are used. Therefore, the resolution is limited by the memory of the GPUs which are used for the training. For being able to achieve high resolutions, e.g. octree structures [131, 142, 105] or hashing-based approaches [74, 119] are used which are able to represent the data in a more memory efficient way. Since these types of approaches are relevant for the development of our solution for the *Manufacturability Analysis* for AM, we will explain it in more detail in Chapter 6.

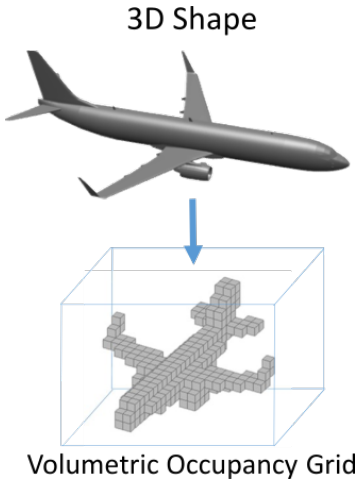


Fig. 4.16 Example of a structured voxel grid [103].

Image-based

As explained above, 3D data-based approaches are limited mainly by their memory requirements. The advantage that they can capture all geometric information of the processed 3D

models can only be used if a sufficiently high resolution is available. Since this is only possible to a limited extent with current GPUs, less memory intensive projection-based approaches are often used. Instead of using 3D data directly, 2D projections, e.g. in the form of images, are used to solve the task at hand. A drawback of 2D-based approaches is that during the transformation of 3D data to 2D data, parts of the information of the original 3D data is lost. To compensate this disadvantage, the first multi-view-based approach Multi-view Convolutional Neural Network (MVCNN) was developed in 2015 [126]. The basic structure of a multi-view based approach is shown in Figure 4.17. The used 3D models are observed from e.g. 12 different angles and processed in parallel. Thus, the loss of information caused by the 2D projection can be compensated at least partially.

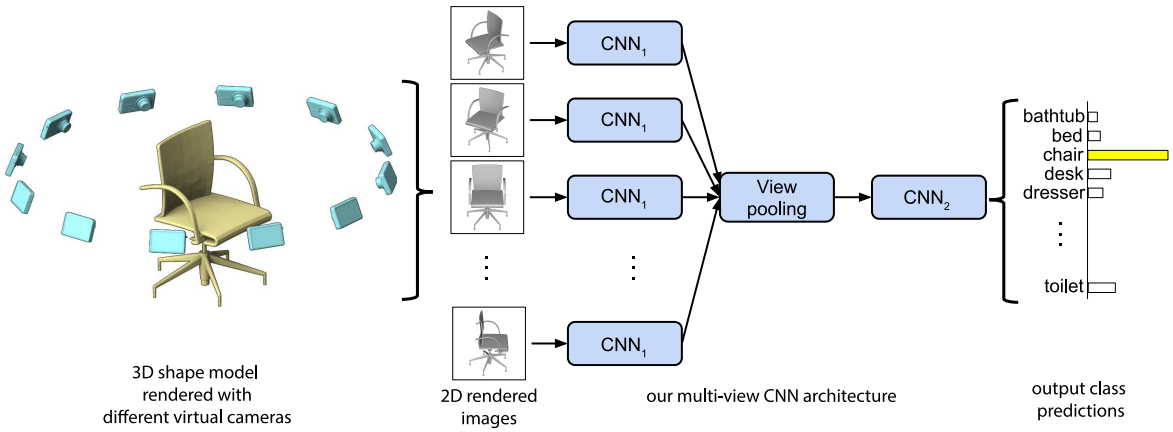


Fig. 4.17 The basic structure of a multi-view CNN [126].

Based on MVCNN [126], several new multi-view architectures have been developed. Currently, the RotationNet approach achieves the state-of-the-art accuracy in the Princeton ModelNet40 classification challenge with an accuracy 97.37 % [56]. Multi-view approaches are therefore able to achieve the best results for tasks in the field of 3D object recognition or segmentation even though there is a loss of information compared to 3D data-based approaches. However, there is a disadvantage if 3D models have to be analyzed as a whole including internal structures. The 2D projections can not show internal structures so that crucial information may not be available for a specific task. Due to their state-of-the-art performance in object classification tasks, especially multi-view-based approaches offer very promising possibilities for being used for the process step of *3D Component Recognition* of additively manufactured components.

The approaches described above form the basis for the development of our solutions for the *Manufacturability Analysis* for AM and *3D Component Recognition* of additively manufactured parts. In Chapter 5, we will explain existing solutions for *Manufacturability*

Analysis and *3D Component Recognition* respectively solutions for similar tasks which are relevant for this thesis, too. Based on the combined information of the Chapters 3, 4 and 5, we are subsequently describing the development of the solutions for the two process steps *Manufacturability Analysis* and *3D Component Recognition* in the Chapters 6 and 7.

Chapter 5

Related Work

In Chapter 3, we have described the current status and emerging problems of the two process steps *Manufacturability Analysis* and *3D Component Recognition*. Subsequently, in Chapter 4, we provided an overview of ML, DL and 3D DL approaches which represent the basis for possible solution concepts for both problems. In this chapter, we summarize which solution approaches for an automation of the two process steps of *Manufacturability Analysis* and *3D Component Recognition* already exist. In addition, we give an overview of innovative solution approaches for similar problems from other domains. In order to classify the strengths and weaknesses of the individual approaches, they are evaluated based on the requirements defined in Chapter 3. Based on this information, we are able to develop our own solution approaches and tailor them optimally to the two problems of *Manufacturability Analysis* and *3D Component Recognition*.

5.1 Manufacturability Analysis

As explained in Chapter 3, the *Manufacturability Analysis* for AM is a complex problem since the causes of restrictions are complex in nature and, accordingly, the capturing of restrictions in the form of rules is not trivial. The developed design rules serve mainly as guidelines for engineers who want to design new components directly in an AM-compatible way. In many situations, however, it is necessary to be able to check existing 3D models with regard to their manufacturability. This is the case for the daily processes at AM service providers. For this reason, different approaches have been developed in recent years to evaluate the manufacturability of already existing 3D models. The existing approaches can be divided into the two categories of design rule-based and simulation-based approaches.

5.1.1 Design Rule-based Analysis for AM

As explained in Section 3.1.2, currently existing industrial software solutions are only able to verify the design rule for *minimum wall thickness*. Additionally, the results often have relatively poor quality. Since the current solutions are not satisfactory, various researchers have been working on the development of alternative approaches for an automated *Manufacturability Analysis* based on the existing design rule catalogs. As already described in Section 3.1, the design rules which have been developed by several researchers, are mostly based on basic geometric elements. Since the design rules, as described in Section 3.1, can not be applied directly to existing complex geometries, several researchers have developed approaches for the application of the design rules for the evaluation of existing geometries. These approaches can basically be divided with regard to the data type they are using for the evaluation. Most of the different approaches are either directly using the 3D meshes or a voxel representation [150]. In this section, we will present an overview of the developed approaches.

Rudolph and Emmelmann developed a tool for an automated verification of parts of the design rules from the design rule catalog which has been developed by Kranz and Emmelmann [62], directly on the basis of the mesh structure of the STL models [109, 110]. They developed their tool for a web platform-based ordering process. From the set of existing design rules, they have chosen the rules for *maximum part dimensions*, *minimum wall thickness*, *gap dimensions*, *cylinder diameters* and *bore diameters*. They have chosen this subset of rules because these rules are independent from the orientation of the component to be produced and allow an automated check directly based on the STL file type. The first restriction is important because the build orientation is not known at the time of the 3D model upload [109].

The maximum part dimensions can be checked relatively easy. The 3D model which is investigated, is rotated around the X-, Y- and Z-axis in fixed steps. For each orientation, it is verified if the 3D model fits into the build chamber of a considered AM machine or not. As soon as a valid orientation is found, the verification is finished. If no valid orientation is found, the part is too large for the AM machine. The verification of the rules for *minimum wall thickness*, *gap dimensions*, *cylinder and bore diameters* are more complex since the corresponding features have to be detected in the STL files first, for being able to check their dimensions subsequently. Walls and gaps can be detected using the normal vectors of the triangles which are the basis of the STL format (See Section 2.1.1). When the normal vectors of two opposite triangles are pointing away from each other, these triangles belong to a wall, if they point towards each other, they belong to a gap.

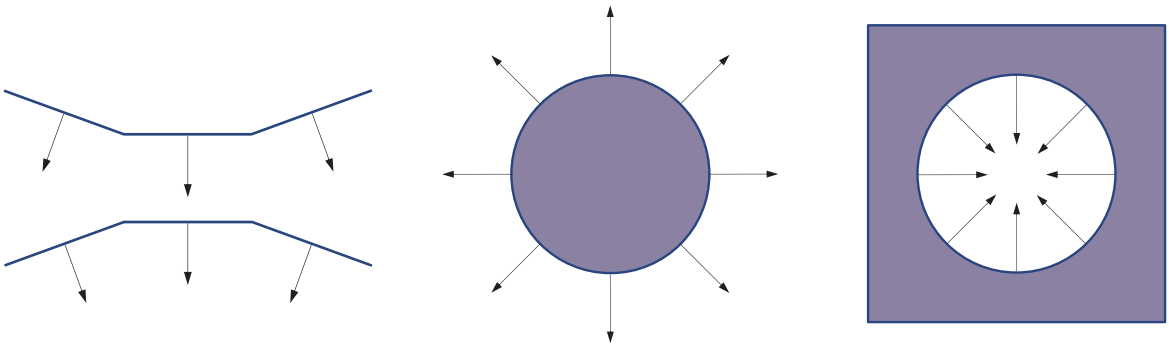


Fig. 5.1 Bores and cylinders can be detected by searching for concave and convex structures [109].

Cylinders and bores can be found the same way as walls and gaps are found. In principle, a cylinder is a concave wall. Concave, respectively convex areas in the STL file can again be found using the normal vectors of the triangles (See Figure 5.1 left). For convex structures (Figure 5.1 left top), the normal vectors of adjacent triangles diverge, for concave structures they converge (Figure 5.1 left bottom). Using that method, walls and bores can be found (See Figure 5.1 center and right). In principle, for the detection of these features, each triangle of a STL model would have to be compared with all other triangles, what leads to extreme computation times for complex STL models. By limiting the search space to all triangles in the neighborhood, the search time can be reduced. Despite this optimized feature search, the developed algorithm for the detection of bores and cylinders needs about 200 seconds for STL models with about 40000 triangles [109]. 3D models ordered at AM service providers often consist of significantly more triangles [101].

Subsequent to the detection of the features, their manufacturability can easily be checked using the geometric thresholds for each case. Additionally, the detected critical features can be visualized by coloring of the corresponding triangles [109].

Shi et al. use a feature-based approach for the recognition of a subset of features from the existing design rule catalogs, too. They are focusing on the features *unsupported features* (previously called overhangs in Section 3.1), *minimum feature size*, *vertical aspect ratio*, *minimum spacing* (small cavities) and *minimum self-supporting angle* [122]. For the detection of these features, they are using the Heat Kernel Signature (HKS) shape descriptor. The HKS shape descriptor is able to describe and therefore also detect geometries based on their heat diffusion characteristics. Basically, different geometries have different heat diffusion behaviour and can therefore be distinguished using the HKS descriptor [128].

The authors tested their method on sample parts which have been designed to demonstrate the detection of the five chosen features *unsupported features*, *minimum feature size*, *vertical aspect ratio*, *minimum spacing* and *minimum self-supporting angle*. The general work flow

is shown in Figure 5.2 on the example of cylinders. In the first step the cylinders are detected using the HKS feature descriptor. Subsequently, the minimum axes is searched and based on that dimension the *minimum feature size* is validated. On the bottom left of Figure 5.2, the visualization of critical geometries can be seen.

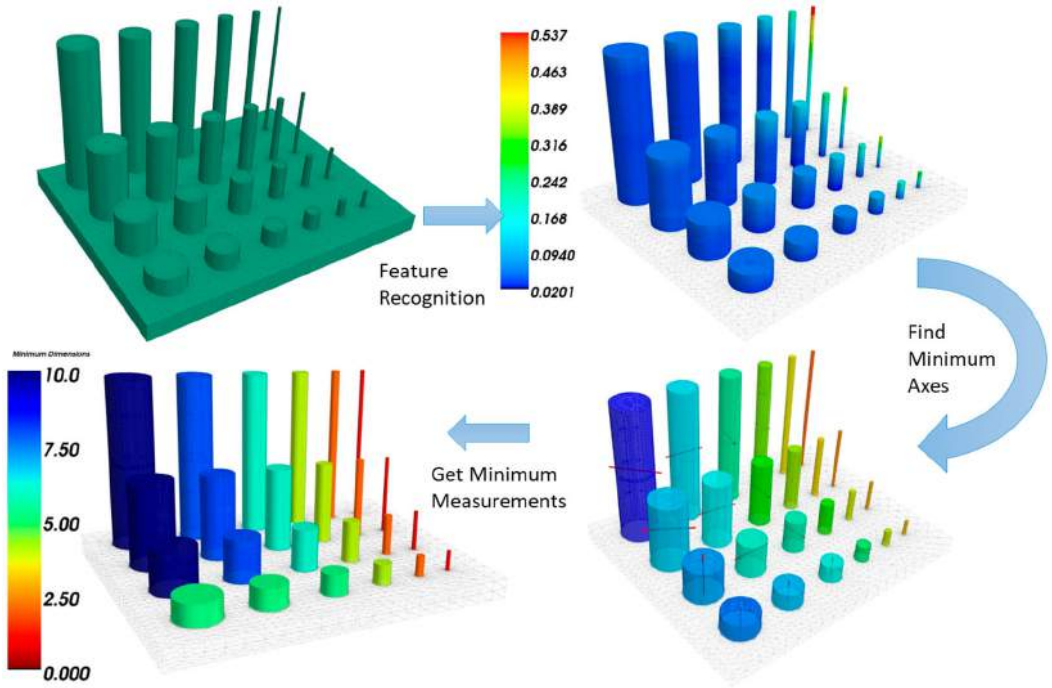


Fig. 5.2 Work flow of the manufacturability analysis using HKS [122].

The authors have shown that their method is able to detect the features in sample geometries. Nevertheless, to the best of our knowledge, the approach has not been evaluated on more complex geometries which occur in the daily production of AM service providers. In principle, the approach can also learn other features by using the HKS kernel and corresponding example geometries. One problem with this approach, however, is that the creation and recognition of features using the HKS descriptor and subsequent evaluation of the manufacturability is designed for design rules which can be evaluated on a parametric basis. According to the design rules, only basic geometries, which can be described parametrically, can be evaluated. The adaptation for complex manufacturability criteria is therefore not possible.

Various other research approaches process the 3D models in the form of voxel models [133, 134, 93, 140]. Tedia and Williams have developed a voxel-based approach for the detection of small features and small holes [133]. They first use a voxelization algorithm to create a voxel representation of the original STL model. Based on that representation, they are able to measure the thickness of a 3D model from multiple directions using ray

Table 5.1 Ranking of the currently used industrial tools and the design rule-based research approaches for *Manufacturability Analysis*.

Method/Metric	R_{1-ma} Adaptability	R_{2-ma} Accuracy	R_{3-ma} Suitable Data Representation	R_{4-ma} Visual Feedback	R_{5-ma} Computation Time
Industrial Tools	– –	0	0	+	0
Rudolph and Emmelmann	– –	No statement possible	+	+	– –
Shi et al.	0	No statement possible	+	+	No statement possible
Tedia and Williams	– –	No statement possible	+	+	–

tracing from the corresponding directs. By ray tracing, the intersection points of the ray and the voxel model can be calculated. Based on these intersection points, small features as well as small holes can be detected. The approach is able to approximate the original STL models using voxel models with a resolution of up to 1000^3 voxels. The voxelization process takes about a minute for 3D models consisting of 10000 to 60000 facets. Based on this representation, their algorithm is able to detect the features.

The different approaches described above have different advantages and disadvantages. We evaluate the approaches in relation to the criteria $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$, which we defined in Section 3.1.3 in order to estimate which properties may be useful for our own solution. Table 5.1 shows the ranking of the different approaches including the industrial software tools. The rating for the industrial tools has already been provided in Section 3.1.3 and is listed again here for direct comparison.

The approaches of Rudolph and Emmelmann [109] and the voxel-based approach of Tedia and Williams [133, 134, 93, 140] have the major disadvantage that they are limited to a small subset of design rules. Due to their implementation characteristics, it is difficult to extend them for the evaluation of further design rules. We therefore rate the performance of the approach of Rudolph and Emmelmann and the approach of Tedia and Williams as insufficient for our needs regarding the requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$. In contrast, the approach of Shi et al. [122] in general offers an extensible structure. Nevertheless, this approach is also limited to geometric properties that can be described based on standard parametric geometries. The performance of the approach of Shi et al. regarding the metric $R_{1-ma} \mathrel{\widehat{=}} adaptability$ is therefore rated as fair.

Based on the publication of the individual approaches, no statement can be made regarding the requirement $R_{2-ma} \mathrel{\widehat{=}} accuracy$, as no statistical tests were carried out using benchmark data sets.

The different data representations which are used by the different approaches again offer potentials but also have drawbacks. Rudolph and Emmelmann are directly using the triangle-based surface representation of the 3D models. This representation is in principle accurate but leads to a high computation time for their evaluation, if the resolution is high. The same is valid for the approach of Tedia and Williams. With a high resolution voxel representation, they are able to represent the 3D models with a high degree of details. However, this is also accompanied by a high computation time for checking the manufacturability criteria. However, this weakness arises from the combination of the chosen data structure with the particular evaluation approach. Since the data representations basically allow a relatively detailed representation of the 3D models, we rate the quality of the approach of Rudolph and Emmelmann and the approach of Tedia and Williams regarding the requirement R_{3-ma} $\mathrel{\widehat{=}}$ *suitable data representation* as good. The HKS descriptor used by Shi et al. is able to represent complex geometries and distinguish between different geometries with high a discriminative quality. Since only local criteria are verified in the experiments described in their publication [122], no statement can be made about the extent to which the HKS kernel can be used to detect global manufacturability criteria. Since HKS kernels are in principle able to describe local and global geometric properties [21], we rate the data representation as good.

A positive aspect of all approaches described above is that they provide a visual feedback based on their analysis of the design rules which is relatively precise. A small drawback of the visual feedback of the approach of Rudolph and Emmelmann is that the quality of their feedback is related to the resolution of the mesh representation of the 3D models. When the mesh has a low resolution, the visual feedback becomes inaccurate. We therefore rate the quality regarding the requirement $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ as good for the approach of Rudolph and Emmelmann. The approach of Shi et al. offers a good visualization for detected critical features. In addition to simply marking the critical areas, the extent to which a criterion has been exceeded can also be visualized using different colors. However, the visualization is limited by the fact that it can only be generated for the standard geometries which the approach is using for the definition of the critical geometries. The visualization of the approach of Shi et al. is therefore rated as good. Due to the high resolution of the voxels used by the approach of Tedia and Williams the feedback, which is generated, offers a high resolution, too. Nevertheless, the generated feedback only offers the option to mark a voxel as critical or non-critical. Marginal areas can not be visualized. We therefore rate the quality

of the approach of Tedia and Williams regarding the requirement $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ as good.

The approach of Rudolph and Emmelmann is relatively computationally intensive. The computation time is already in the range of several minutes for 3D models with low complexity. Since the computation time is directly dependent on the amount of triangles of the 3D models, it increases with their complexity. Therefore, we rate the performance of the approach of Rudolph and Emmelmann as insufficient for our needs regarding the requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$. The voxelization of 3D models for the approach of Tedia and Williams also takes more than a minute in the examples presented in their publication. However, this value is not directly related to the resolution of the 3D models, but to the resolution of the voxel grid and therefore does not necessarily increase with the complexity of the initial 3D models. Since the calculation of the voxel representation often takes more than one minute, we rate the quality of the approach in terms of the $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ requirement as inadequate. For the approach of Shi et al. it is unclear how much computation time is required, based on the information presented in their publication.

5.1.2 Simulation-based Analysis for AM

In the previous Section 5.1.1, we described design rule-based approaches for an automated *Manufacturability Analysis*. As already explained in Section 3.1, the design rules are intended to describe the partially complex constraints generated by e.g. thermal processes during the manufacturing process. Subsequently, these rules are evaluated based on different data representations. In contrast to the rule-based approaches, the simulation-based approaches bypass this intermediate step. With the help of simulations, the complex thermal processes are to be directly simulated and thus potentially non-manufacturable geometries are to be identified.

Various ready-to-use software solutions are already available for the simulation-based evaluation of the manufacturability of 3D models [125, 9, 124]. They are designed for the analysis of complex high performance metal components, as production errors for these components are extremely costly and must therefore be avoided.

The existing tools are able to predict distortions or thermal stresses and the corresponding cracks using the Finite Element Method (FEM). For the FEM analysis, the 3D model is meshed first, i.e. the 3D model is represented by a finite set of sub-body elements. Subsequently, the physical behavior of these simple elements can easily be simulated. By linking the behavior of the individual elements, the behavior of the entire body can be simulated [154].

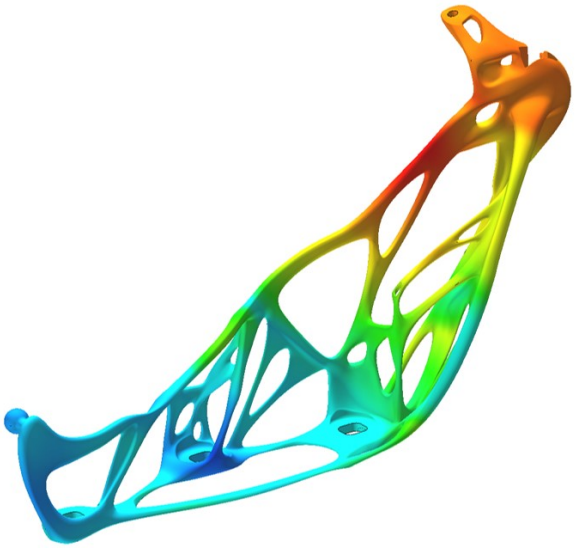


Fig. 5.3 Example of a displacement analysis using the FEM [125].

An example of a displacement simulation can be seen in Figure 5.3 [125]. The simulated displacement is growing from blue over green to red. These software tools offer a powerful solution for the simulation of complex manufacturing constraints. The results are close to the real physical behaviour and can therefore predict possible production failures with a high accuracy based on a precise representation when using finite elements with a sufficiently small size. Additionally, a precise feedback is generated.

Nevertheless, a major drawback of the simulation-based approaches is that the FEM simulation takes from several minutes to several hours. For the use-case of simulating the production process of expensive and complex metal components, it is a valuable solution. Nevertheless, for our use-case of *Manufacturability Analysis* in the process chain of AM service providers, the analysis has to be finished within a few seconds or a few minutes at most for being able to provide real-time feedback. Therefore, we rate the performance of simulation-based approaches regarding the requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ as insufficient. Since this is a crucial criterion, simulation-based approaches can not be used for our purpose.

5.1.3 Machine Learning-based Analysis

In the previous sections, we have described existing approaches for *Manufacturability Analysis* in the AM domain. Besides the approaches above, there are also approaches from other manufacturing domains. In this section, we will explain an DL-based *Manufacturability Analysis* approach from a different domain that is relevant for our further work.

Table 5.2 Ranking of the currently used industrial tools, the design rule-based research approaches and simulation-based approaches for *Manufacturability Analysis*.

Method/Metric	R_{1-ma} Adaptability	R_{2-ma} Accuracy	R_{3-ma} Suitable Data Representation	R_{4-ma} Visual Feedback	R_{5-ma} Computation Time
Industrial Tools	– –	0	0	+	0
Rudolph and Emmelmann	– –	No statement possible	+	+	– –
Shi et al.	0	No statement possible	+	+	No statement possible
Tedia and Williams	– –	No statement possible	+	+	–
Simulation-based	+	++	++	++	– –

Balu et al. developed an DL-based approach for the *Manufactuability Analysis* of drilled holes [15, 14, 30]. Similar to the AM domain, several rules for the drilling of bores exist which describe non-manufacturable geometries. Possible problems are caused by too thin walls at the edges of bores or a critical ratio of depth to diameter of a bore. The basic idea of the developed approach is that an DL model learns which geometries are manufacturable and which are not, based on existing labeled geometries which are processed using a voxel representation [15].

The basic structure of the approach is shown in Figure 5.4. The authors have chosen to use a 3D-CNN as central element of their approach. A 3D-CNN is able to process 3D models in form of a voxel representation. Therefore, the first step of the work flow shown in Figure 5.4 is the voxelization of the CAD models. Subsequently, the 3D-CNN is able to process the voxel models. Based on a data set of labeled 3D models, the system is able to learn which geometries are manufacturable and which are not. In addition to the pure decision regarding the manufacturability of a 3D model, the approach also generates visual feedback. For that purpose, the authors added the 3D-Grad-CAM algorithm which we explained in Section 4.3.2 [30].

For the training process, an artificial set of 3D models was created using a 3D modeling kernel. The set consists of cuboids which contain bores at varying positions with varying parameters for the diameter and the depth of the bore. With that procedure a set of 9531 3D models which include manufacturable and non-manufacturable 3D models was created. For the subsequent tests, 675 additional 3D models have been created. All of these 3D models have been voxelized with a resolution of 64^3 voxels. The voxel representation was created in

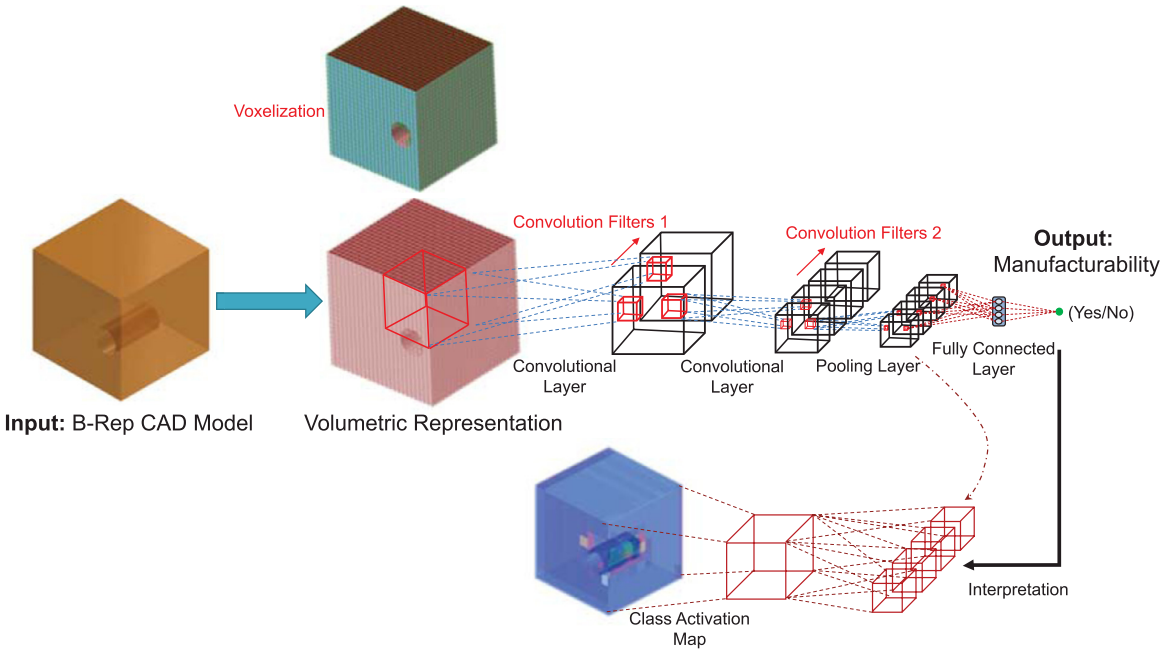


Fig. 5.4 Structure of the ML-based *Manufacturability Analysis* approach of Balu et al. [30].

two variants. In the first variant, all voxels inside the 3D model are represented by a 1, all outside voxels by a 0. In a second variant, the normal vector of the surface is additionally stored for each voxel lying on the surface of the 3D model. This vector is normalized and described by its X-, Y- and Z-value [15].

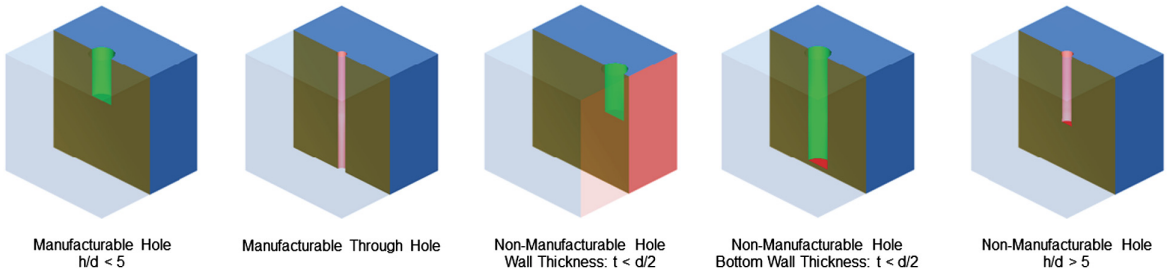
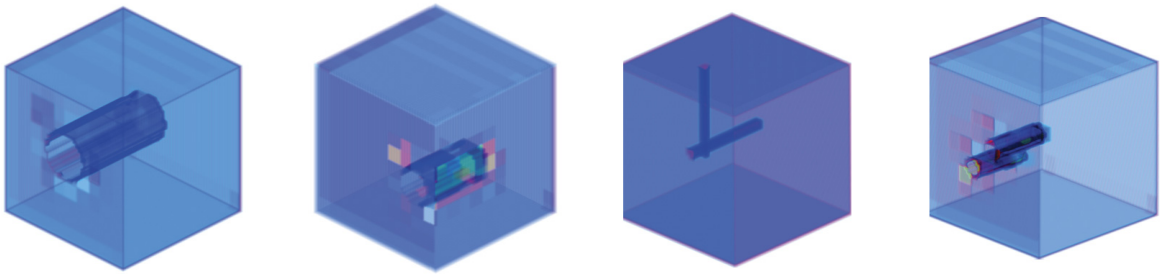


Fig. 5.5 Examples of 3D models from the training data set [30].

Based on the described training data, the 3D-CNN is trained. In the subsequent evaluation, the approach achieves an classification accuracy of 79.52% correctly evaluated 3D models for the second variant which uses the additional information of the normal vectors. Without normal vectors, an accuracy of 71.36% is achieved [15].

Examples of the 3D-Grad-CAM visualization can be seen in Figure 5.6 [30]. As can be seen, the algorithm marks the critical areas within the 3D models. However, the representation is relatively indistinct.



(a) Manufacturable (b) Non-Manufacturable (c) Non-Manufacturable (d) Non-Manufacturable

Fig. 5.6 Examples of the 3D-Grad-CAM visualization [30].

Overall, the approach offers the possibility to learn manufacturability criteria based on appropriately labeled data sets. Based on the described results, we rate the quality of the approach regarding the requirement $R_{2-ma} \mathrel{\widehat{=}} accuracy$ as good. Due to its data driven approach, it is easily extendable to other design criteria if the respective data is available. We therefore rate the performance of the approach with respect to the requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$ as excellent. The computation time is not described in the corresponding publications. In general, however, one benefit of NNs is that although the training phase is very computationally intensive, the subsequent application can usually be carried out almost in real time. The performance of the approach with respect to the requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ is therefore rated as good. The resolution of the approach is limited to 64^3 voxels using current GPUs. This resolution is inadequate for our purposes. For being able to investigate more filigree features, a higher resolution would be necessary. Also the visualization approach is basically promising, but in its current form it is indistinct and therefore rated as inadequate.

In this section, we provided an overview of methods for *Manufacturability Analysis* in the AM domain and the related domain of bore drilling. The approaches were evaluated with respect to the requirements $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ defined in Section 3.1.3. An overview is shown in Table 5.3.

As already described above, the simulation-based approaches offer a reliable and accurate simulation of the physical behaviour of the build process and can therefore predict possible build errors with a high reliability. Nevertheless, the simulation is a complex process and often takes several hours. This process duration is definitely an exclusion criterion for our application.

For being able to provide a comprehensive *Manufacturability Analysis* solution, the requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$ has a major importance. Therefore, the approaches

Table 5.3 Ranking of the currently used industrial tools, the design rule-based research approaches and simulation-based approaches for *Manufacturability Analysis* in the AM domain and the ML-based approach of Balu et al. from a domain of bore drilling.

Method/Metric	R_{1-ma} Adaptability	R_{2-ma} Accuracy	R_{3-ma} Suitable Data Representation	R_{4-ma} Visual Feedback	R_{5-ma} Computation Time
Industrial Tools	– –	0	0	+	0
Rudolph and Emmelmann	– –	No statement possible	+	+	– –
Shi et al.	0	No statement possible	+	++	No statement possible
Tedia and Williams	– –	No statement possible	+	+	–
Simulation-based	+	++	++	+	– –
Balu et al.	++	+	–	–	+

of Rudolph and Emmelmann or Tedia and Williams are not suitable for a comprehensive *Manufacturability Analysis* because they are limited to a small subset of design rules.

The remaining approaches of Shi et al. and Balu et al. are both using features for the analysis of the manufacturability of 3D models. The difference lies in the nature of the features. Shi et al. are using the fixed HKS feature descriptor, to detect predefined standard geometries and subsequently evaluate the detected areas using the parameters describing the predefined geometries. Therefore, the approach is limited to design rules which can be described by combinations of standard geometries, too. The approach of Balu et al. is based on self-learned deep features of the CNN architecture. This enables the approach to be adapted to further criteria using correspondingly labeled data sets. Since the usage of the deep features leads to a good adaptability, we believe that it is more promising to apply a DL-based approach compared to the HKS-based approach for the *Manufacturability Analysis* for AM.

Besides the positive aspect of high adaptability of the approach of Balu et al., it offers the important benefit of a short computation time. The main disadvantage of the approach is the very coarse data representation. However, in Section 4.4, we have shown that voxel-based approaches exist that can achieve much higher resolutions. We are therefore confident that a higher resolution enables the development of a 3D-CNN-based solution, which is able to learn to evaluate 3D models regarding their manufacturability for AM. The currently very imprecise visualization of critical geometries is the biggest drawback of the approach of Balu

et al.. Nevertheless, in Section 4.3.3 we presented additional methods for the generation of visual feedback which provide explainable behaviour for DNNs. Based on these and using a higher voxel-grid resolution, we are confident that we can generate accurate feedback.

Based on the information provided in this chapter, we have decided that a voxel-based 3D-CNN approach for assessing the *Manufacturability Analysis* for AM is the most promising approach. Based on the general idea of Balu et al.'s approach, we want to compensate its weaknesses by the alternative methods described in Chapter 4 and develop a solution which is optimized with regard to the defined requirements. The exact structure of our solution will be described in detail in Chapter 6.

5.2 3D Object Recognition

For being able to develop an appropriate solution for the issue of *3D Component Recognition* of additively manufactured parts, we first need an overview regarding existing approaches for this task respectively similar tasks. In Section 4.4, we gave an overview about the basic approaches for traditional and ML-based computer vision. These approaches form the basis for the development of solutions for different real-world applications. In recent years, *3D Object Recognition* has developed into an increasingly important task, as automation solutions requiring object recognition are entering more and more areas of everyday life and industry. Automated *3D Object Recognition* is used for autonomous driving, robotic grasping or the observation of construction applications [139, 77, 24]. Due to the increasing interest from the different domains, the research and development in the field of *3D Object Recognition* was also increased and more and more different approaches were developed. In this section, we will first give a short overview regarding object recognition in several real-world applications and subsequently introduce an industrial approach for *3D Component Recognition* of additively manufactured parts which recently emerged. Subsequently, we will discuss the positive and negative aspects of the different approaches regarding the development of our own solution for *3D Component Recognition* of additively manufactured parts.

5.2.1 Object Recognition in Real-World Applications

Nowadays, object recognition systems are used in many real-world applications like autonomous driving or industrial assembly lines. In this section, we will give an overview about existing applications which are related to our task of *3D Component Recognition* of additively manufactured parts.

Autonomous Driving

The recognition and tracking of objects like pedestrians, other vehicles, obstacles or street signs in the environment of autonomous vehicles is one of the most important tasks in the domain of autonomous driving. In order to enable autonomous driving, real-time solutions are needed that are able to recognize the described objects quickly and reliably. Nowadays, CNNs, like the architectures described in Chapter 4, are often used for this purpose. The CNNs can be trained on image databases like the Caltech-101 database [29] which consists of images of 101 different object categories like pedestrians, bikes or cars. Trained state-of-the-art CNN-architectures reach recognition accuracies of more than 90% for the recognition of the objects of the Caltech-101 data set [139].

Robotic Grasping

Besides the application for autonomous driving, object recognition systems are also often used in industrial applications. One of the most common applications is robotic grasping. In modern assembly lines, robot arms are often used to grasp the objects which are processed in the assembly line and e.g. pass them to the next assembly steps. For being able to grasp the objects fully automated, the robotic systems needs to recognize the objects using some type of sensors which generate e.g. images or point clouds. Based on the sensor data, the system has to be able to recognize the objects and calculate the corresponding poses [79].

Since most assembly lines are relatively static systems, created for a consistent production, the object recognition systems which are used, have to be designed for the detection of a fixed set of components. Luo and Kuo [79] developed a system to recognize and grasp a fixed set of objects using geometry-based descriptors. Their system is able to recognize six different objects with an accuracy of about 98%.

5.2.2 3D Component Recognition in the AM Domain

In November 2018, when we already started our work on the *3D Component Recognition* of additively manufactured parts, an industrial solution for the task emerged on the market. The company AM-Flow has developed an DL-based system, specialized for the recognition of components produced using the SLS process. The system is using a multi-view image-based DL architecture which is able to recognize produced components based on images taken from four different cameras [7].

To the best of our knowledge, the system has not been evaluated on any benchmark data sets, yet. According to their own statements, the system achieves an accuracy of 95 percent correctly recognized components with zero percent false recognitions. The system

is able to recognize about 20 components per second [6]. With their solution, the AM-Flow company has shown that it is possible to create an DL-based solution for the *3D Component Recognition* of additively manufactured parts which can achieve a high accuracy. Nevertheless, there is still a small gap to a 100 percent accuracy which has to be closed.

In Section 4.4, we have discussed several general approaches for object recognition task. Subsequently, in this section, we gave an overview regarding existing applications of object recognition approaches for real-world tasks. In conclusion, we want to evaluate the advantages and disadvantages of the different approaches with respect to the requirements $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{3-cr} \mathrel{\widehat{=}} adaptability$, $R_{4-cr} \mathrel{\widehat{=}} process\ time$, $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$ defined in Section 3.2.2 for the application of *3D Component Recognition* of additively manufactured parts in order to be able to design our own approach based on this information.

In Section 3.2.2, we have explained that manual processing of the *3D Component Recognition* step is not practical and will not be feasible anymore with growing production volumes. In order to be able to replace the manual approach with an automated solution, two main groups of solution approaches are available: Systems based on traditional CV and systems using DL-based CV. We have summarized the advantages and disadvantages of the two groups of approaches with regard to the requirements defined in Section 3.2.2 in Table 5.4.

Our rating regarding the metrics $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$ are equal for traditional CV approaches as well as for DL-based CV approaches. Both groups of solutions are in principle capable to detect objects with a high accuracy and robustness, if they are set up properly. The costs and also the scalability are mostly influenced by the hardware which is used. That is basically the same for traditional CV approaches as well as ML-based CV approaches since they are using the same types of sensor data. Since the use of e.g. camera-based systems does not require very expensive hardware, a possible image-based solution system can be scaled without disproportionate costs.

Major differences between traditional CV approaches and DL-based CV approaches become apparent when considering the two requirements $R_{3-cr} \mathrel{\widehat{=}} adaptability$ and $R_{4-cr} \mathrel{\widehat{=}} process\ time$. Traditional CV methods are mainly used for tasks where a fixed set of different objects has to be recognized. In Section 4.4.1, we have explained that traditional CV systems are optimized for a specific task e.g. the recognition of a specific set of different object. The system is designed in such a way that it can optimally distinguish the specific objects of the fixed data set. As explained in Section 3.2.2, the set of components to be recognized is varying everyday at AM service providers. Therefore, a solution has

Table 5.4 Ranking of the manual process, traditional CV approaches and DL-based CV approaches regarding the six metrics recognition rate, robustness, adaptability, process time, scalability, and process costs.

Method/Metric	R_{1-cr} Recognition Rate	R_{2-cr} Robustness	R_{3-cr} Adaptability	R_{4-cr} Process Time	R_{5-cr} Scalability	R_{6-cr} Process Costs
Manual process	++	++	+	−−	−	−−
Traditional CV approaches	++	++	−	−−	+	+
DL-based CV approaches	++	++	++	+	+	+

to be able to adapt to these changes automatically. This is not possible in an automated way with traditional CV approaches. Therefore, it cannot be guaranteed that the system, which was designed to recognize a fixed set of objects, is also able to recognize the daily changing components optimally. We therefore rate the quality of traditional CV approaches with regard to the requirement $R_{3-cr} \mathrel{\widehat{=}} adaptability$ as inadequate. In contrast, the main advantage of DL-based approaches is that they are extremely adaptable due to their data-driven approach. With the help of, for example, images of the daily varying components, they are able to learn the features with the highest discriminative information based on the corresponding images autonomously. This enables them to adjust every day in such a way that the corresponding components can be recognized optimally. The performance of DL-based approaches regarding the metric $R_{3-cr} \mathrel{\widehat{=}} adaptability$ is therefore rated as excellent.

Additionally, the process time for the actual recognition is very short, since the main computational effort is necessary in the training phase of those systems. The computational cost for the inference, on the other hand, is usually very short for DL systems. We therefore rate the quality of DL-based systems with regard to the metric $R_{4-cr} \mathrel{\widehat{=}} process\ time$ as good. In contrast, as explained in Section 4.4.1, a drawback of traditional CV approaches is the high computation time. For the recognition of different objects, the feature representation generated from the sensor data of the physical components must be compared with the corresponding feature representation of each of the 3D models, to determine which component is the most likely in the matching step. With an increasing number of different objects, this process leads to a significant increase of the computing time. With regard to the metric $R_{4-cr} \mathrel{\widehat{=}} process\ time$, we therefore rate the performance of traditional CV approaches as insufficient.

Based on the information presented in Section 4.4.2 and the results achieved by the AM-Vision solution [6], we decided that an DL-based solution is most promising for the purpose of *3D Component Recognition* of additively manufactured parts. As explained in Section 4.4.2, the group of DL-based CV approaches includes a wide range of different concepts. Considering the cost requirement, we believe that an image-based approach is the most promising choice for our purposes. Image-based DL approaches can achieve state-of-the-art performance at the Princeton ModelNet recognition challenge [146] despite the loss of information due to the 3D-2D transformation which can be counteracted by using multi-view architectures. Therefore, we will use an MVCNN-based approach as the main component for the development of our solution for the *3D Component Recognition* of additively manufactured parts. We will explain our developed solution architecture in Chapter 7 in detail.

Part IV

Solution

Chapter 6

Solution: Manufacturability Analysis

In Chapter 3, we described the issue of *Manufacturability Analysis* for AM. We outlined the current state of this process step in the process chain of AM service providers and described the physical causes for manufacturability constraints and the attempts to describe the constraints using design rules. In addition, we have defined requirements which a suitable solution for the *Manufacturability Analysis* must meet. Subsequently, in Chapter 5, we gave an overview about existing approaches which provide an automated *Manufacturability Analysis* functionality and rated their advantages and disadvantages regarding the defined requirements. Besides approaches from the AM domain, we additionally explained an DL-based approach for the manufacturability analysis of drilled holes. Based on this information and the foundations of ML and DL, we decided to develop our own DL-based solution which is optimized for the task of automated *Manufacturability Analysis* in the process chain of AM service providers. As the AM service provider which we used as reference mainly uses SLS and SLM machines, the solution is developed for these technologies. In principle, our solution is suitable for all AM manufacturing technologies. Through its data-driven approach, our solution is designed to extract and harness the information regarding the manufacturability of components which is generated in the daily production of AM service providers.

In this chapter, we will explain our solution and the corresponding building blocks. The chapter is split into two parts: The first one, from Section 6.1 to 6.3 is dealing with the general architecture of our solution, the integration into the environment of AM service providers and the corresponding building blocks of the solution. In Section 6.1, we will explain the general structure of the whole system. Subsequently, in the Sections 6.2 and 6.3, we will explain the main building blocks, i.e. the DL model and the feedback generation system in detail. In the second part, we are focusing on the realization details of our solution. In addition to the basic architecture of our solution described from Section 6.1 and 6.3, the

preprocessing of the raw 3D models is of crucial importance. We will discuss this in Section 6.4. Additionally, we have defined different options to for the design of the final classification stage of our solution. These options are described in Section 6.5.

6.1 Solution Architecture

In this section, we want to explain the concept of our solution for an automated data-driven *Manufacturability Analysis* in the process chain of AM service providers. At first, we are explaining the general structure and work flow of the whole system. Subsequently, in the following sections, we will define and explain the architectures chosen for the DL model and the feedback system.

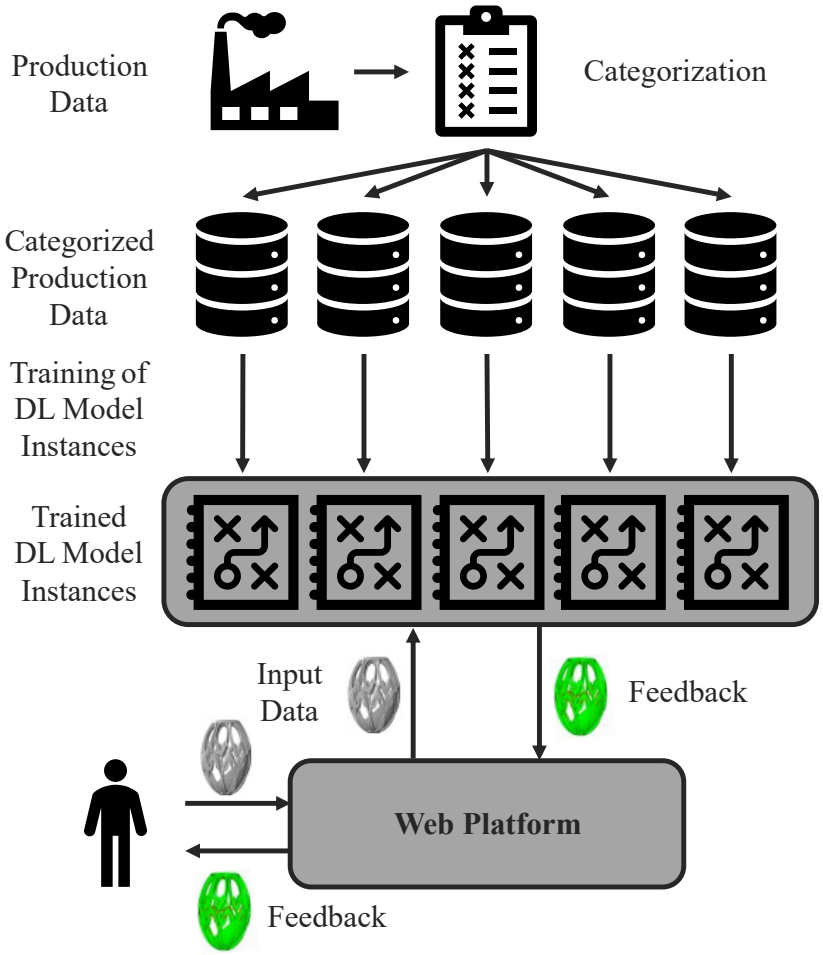


Fig. 6.1 The basic structure of the solution concept.

The basic structure of our solution is shown in Figure 6.1. It is showing the general work flow from extraction and categorization of production data at AM service providers over the

training of separated instances of the DL model for each failure category to the inference phase, where customer 3D models are processed and the feedback consisting of a decision if a 3D model is manufacturable or not and the visual feedback is generated.

Since we want to use a data-driven system for our solution, we need the corresponding data to train the system. The data generated in the production process every day, can be used for this purpose if it is collected and processed accordingly. The production data must therefore first be categorized. Since there are various different causes for production failures, first the various failure categories such as *insufficient wall thickness* or *non-manufacturable bore hole* have been defined based on the expertise of AM engineers. Subsequently, whenever a production failure occurs, the 3D model of the corresponding component can be assigned to one or multiple of the failure categories. This allows a data collection to be generated for each failure category. All components that are manufactured without defects, are classified as manufacturable.

Based on the collected data for each category, a separate instance of the DL model can be trained. On the one hand, due to the separation into individual failure categories, it is easier for the DL model to learn the decisive geometric criteria and on the other hand it is possible to generate more detailed feedback. Since each instance is learning a separate manufacturability criterion, this information can be used to provide the customer with an enhanced feedback. Once the instances of the DL model are trained based on the categorized production data which is used as training data, they are able to evaluate the 3D models of the customers subsequent to their upload. As a result, the customer gets a binary decision if a 3D model is manufacturable or not and if not, additionally a visual feedback.

Our solution consists of two main building blocks shown in Figure 6.2: The DL model which provides the functionality for generating the raw binary decision if a 3D model is manufacturable or not and additionally the feedback system which generates a visualization of critical geometries. In the next sections, we will describe the chosen DL model and the corresponding feedback generation system including their internal procedures in more detail.

6.2 DL model: Spatial Hashing-based CNN

As we have highlighted in Section 5.1, the 3D CNN-based approach of Balu et al. has shown that it has the capability of learning manufacturability criteria in the domain of bore drilling. Based on the results of Balu et al., we therefore decided to use a 3D CNN-based as the main building block for our solution architecture. Since the main drawback of the approach of Balu et al. is that the maximum resolution of the voxel grid is limited to 64^3 voxels using current GPUs, we need an other 3D CNN architecture for our solution. From the group of

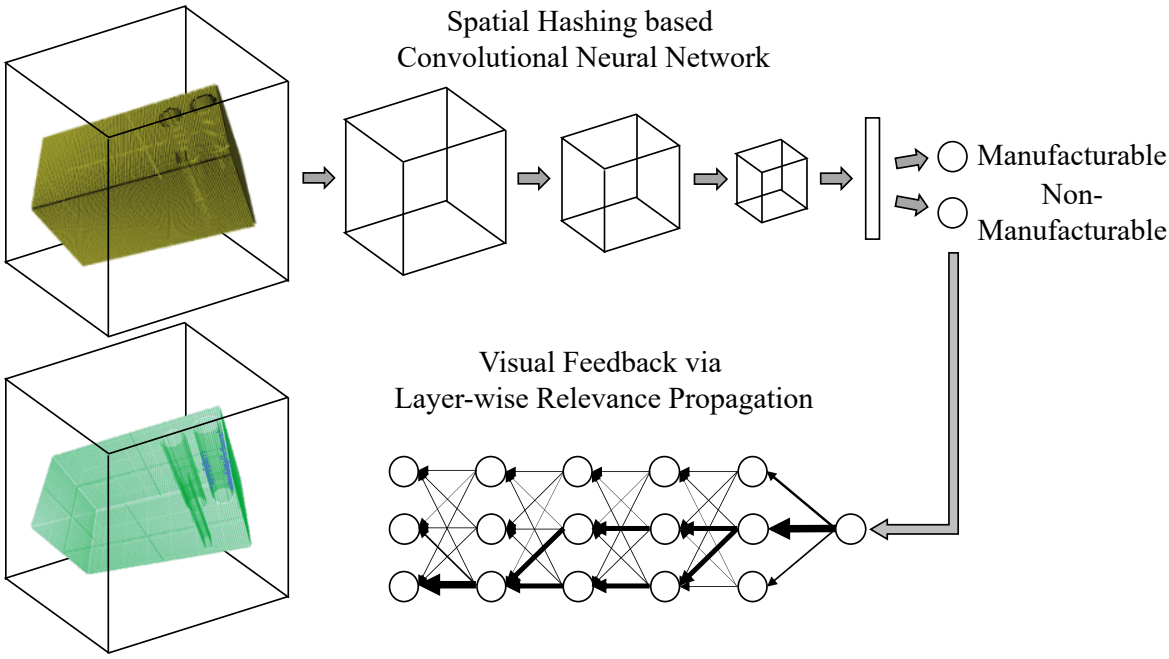


Fig. 6.2 The structure of the chosen DL model and the feedback generation system.

current state-of-the-art 3D CNNs, the HCNN approach is achieving the highest resolution [119]. The high resolution is necessary for being able to represent the 3D models with a high degree of detail for the recognition of filigree critical geometries. For being able to utilize the approach for the task of *Manufacturability Analysis*, several intermediate steps are necessary. In this section, we first describe the structure of the HCNN approach and subsequently discuss its integration into our solution architecture.

As explained above, one major issue is the enormously increasing memory requirement of 3D-CNNs with increasing resolution of the voxel grid. The HCNN approach [119] has dealt with this problem and represents the current state of the art. The approach is based on PSH [74] whereby the voxel models can be stored almost optimally. For storing of a voxel representation with a resolution of N^3 raw voxels, a memory consumption of $\mathbf{O}(N^3)$ would be caused. In contrast, storing the same 3D model using octree-based methods, the memory consumption is bounded by $\mathbf{O}(N^2)$ and the PSH method reaches $\mathbf{O}(N^{\frac{4}{3}})$ [119]. To understand the basic structure of the HCNN approach, we will briefly explain PSH and the corresponding data access for the implementation of the basic CNN functions.

6.2.1 Perfect Spatial Hashing

A perfect spatial hash function is a hash function which has no collisions for a static voxel model. The PSH approach is designed to calculate such functions with the additional

condition of generating the function with minimal memory consumption, i.e. minimal hash- and offset-table size, while optimizing spatial coherence in the hash and offset table for achieving fast data access [74].

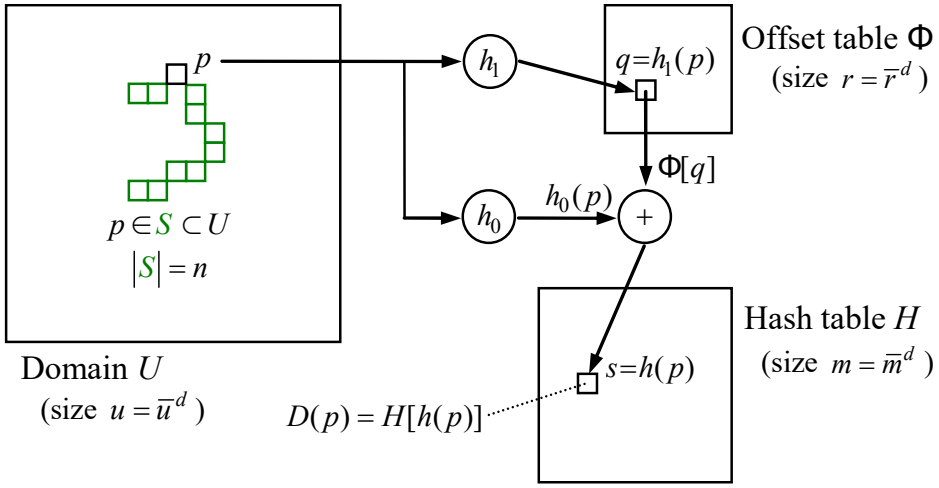
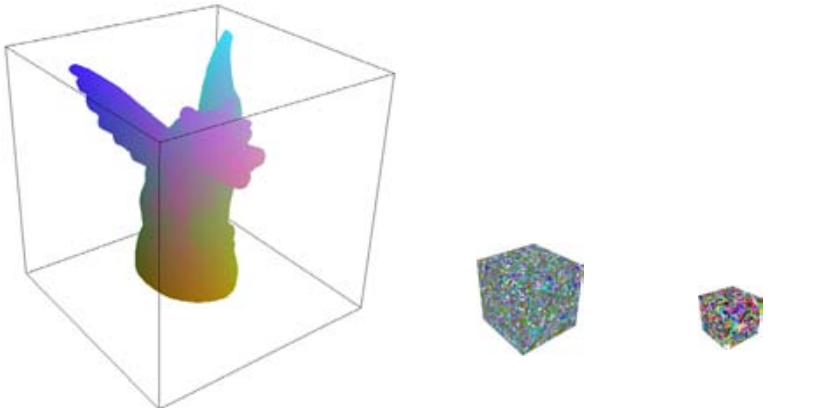


Fig. 6.3 The basic structure of PSH illustrated in 2D [74].

PSH is creating a hash function $h(p)$ for hashing of the discretized model $S \subset U$ which is a subset of the domain U with $u = \bar{u}^d$ positions. S consists of n points p which belong to a data record $D(p)$. In our work, we are dealing with $d = 3$ dimensions. For the explanation of the PSH approach, we use $d = 2$ for a better visualization (see Figure 6.3). The hash function $h(p)$ is used to create a dense representation $H[h(p)]$ of the sparse data $D(p)$.

The perfect spatial hash function $h(p)$ is a combination of two hash functions $h_0(p)$ and $h_1(p)$ (See Equation 6.1). For the creation of the hash-table H, $\bar{m}$ is chosen as the minimum value satisfying the condition $m = \bar{m}^d \geqq n$ so that all n points fit into the hash-table. Both $h_0(p)$ and $h_1(p)$ are designed as simple modulo operators using the respective table sizes $\bar{m}$ and $\bar{r}$. Since $\bar{m}$ is already defined, a minimal value for $\bar{r}$ for the offset table Φ has to be found which creates a perfect overall hash function $h(p)$. This value is found by iteratively increasing $\bar{r}$, testing different values for the offset vector $\Phi[h_1(p)]$ and checking for collisions. If a collision free hash function $h(p)$ is found, the iterative search is terminated. Besides the hash table H, also a position tag table T with the same size as the size of H is created. This table stores the corresponding position p of a voxel in the original dense grid. The position tag table is used to check the validity of a hash entry when trying to load an entry from the hashed data. A detailed explanation is given in Section 6.2.2.

$$h(p) = h_0(p) + \Phi[h_1(p)] \, mod \, \bar{m} \tag{6.1}$$



Sparse 3D data
n=41,127 voxels in 128^3 volume

Hash table H
m=35^3 (=42,875)

Offset table Φ
r=19^3

Fig. 6.4 Example 3D data showing the effect of the efficient data storage [74].

Figure 6.4 shows an example 3D model stored using PSH. Compared to the original data in the dense grid with a resolution of 128^3 voxel, the hash-table and offset-table only require 35^3 respectively 19^3 voxel for storing the same 3D model. That is a reduction to only approximately 2.37% of the original size.

6.2.2 CNN Operations with PSH

For running CNN operations like convolution on the data stored as a PSH representation, the data has to be accessed again. To calculate the convolution for an output voxel p_o, the data in the receptive field of the convolution kernel has to be accessed. Therefore, the data of the input voxel p_i in the center of the convolution kernel (marked in grey in Figure 6.5) and the neighbors in the kernels receptive field have to be loaded. Using that procedure, only the necessary information is loaded from the hashed data for the convolution operation. When trying to load a voxel p_i from the hashed data using the hash function $h(p_i)$, the position tag table is used to confirm that $T[h(p_i)] = p_i$. If that is not the case, an empty voxel which is not part of the surface of the 3D model has been accessed [119].

With the same procedure all other CNN operations can be executed without having the necessity to load the complete dense data. After running an operation, the resulting data can be stored again using the hash functionality.

6.2.3 Memory consumption

Figure 6.6 shows the memory consumption of the HCNN model compared with the Octree-based Convolutional Neural Network (OCNN) architecture [142] which is an octree-based

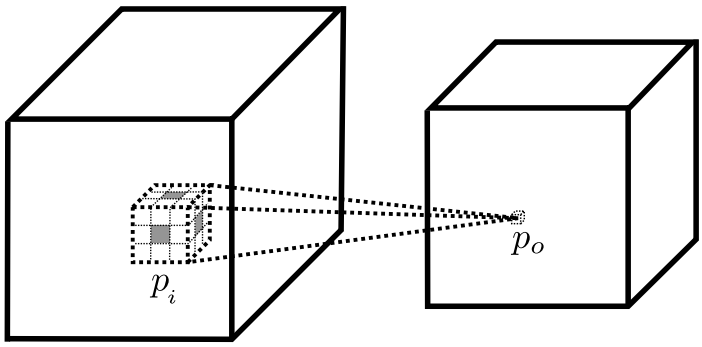


Fig. 6.5 Receptive field of a 3D-CNN convolution kernel.

competitor. The figure shows the memory consumption in mega bytes in relation to the resolution of the voxelized 3D models for NN training with a mini-batch size of 32 models. It can clearly be seen that the HCNN approach can process higher resolutions compared to OCNN. With an NVidia 1080 GTX GPU with 8GB memory, a resolution of up to 512^3 voxel can be reached.

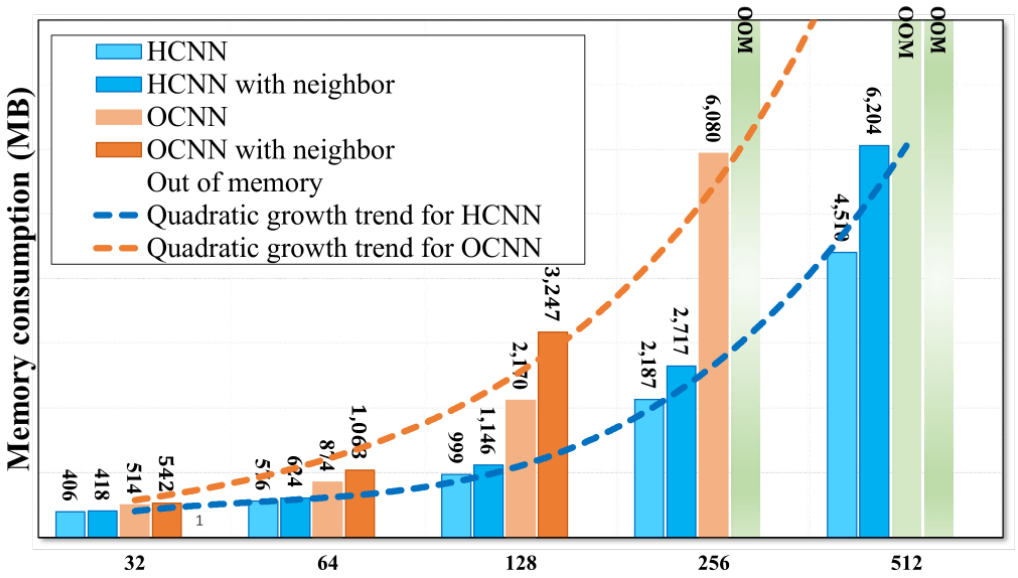


Fig. 6.6 Comparison of memory consumption of HCNN compared to OCNN [119].

6.3 Feedback System

The HCNN model itself can only provide a binary decision if a 3D model is manufacturable or not. The feedback has to be generated by an additional system. In Section 5.1, we have

pointed out that the feedback generation approach used by Balu et al. [15] does not offer feedback with satisfactory quality. The output generated by the 3D Grad-CAM algorithm is very coarsely resolved and vague. In Section 4.3.3, we have outlined that several other methods exist for the generation of visual feedback for image analysis tasks. Based on publications from other fields of application, we have decided that the LRP approach is more promising for our task of *Manufacturability Analysis* than the Grad-CAM approach. In several fields of research, the LRP approach has shown that it is able to generate accurate feedback and to represent the crucial criteria that a DNN has used for its decision with high quality [55, 10, 44, 112, 89]. These qualities are extremely relevant for our solution.

6.3.1 Layerwise Relevance Propagation

The LRP algorithm is one of the decisive components used for our solution. In the last five years, LRP-based approaches have been an essential part of the backpropagation-based group of *Model Explanation* algorithms. Several further works are based on the original publication of Bach et al. [12]: The research group around Bach (now Lapuschkin) itself published various works for further development of their approach [87, 67, 69, 88, 112, 68, 113]. In addition, other researchers have developed modifications of the baseline approach [55, 75] or applied the approach for solving different research tasks [148, 20, 42, 28]. In this section, we will explain the basic functionality in more detail.

The basic idea of LRP is that the critical factors influencing the decision of a NN are propagated back from the output, layer by layer, to the input layer. These critical factors are called *relevance*. The relevance is distributed over the neurons of a layer and propagated back from the following layer by propagation rules (see Figure 6.7) [87].

The LRP-0 rule

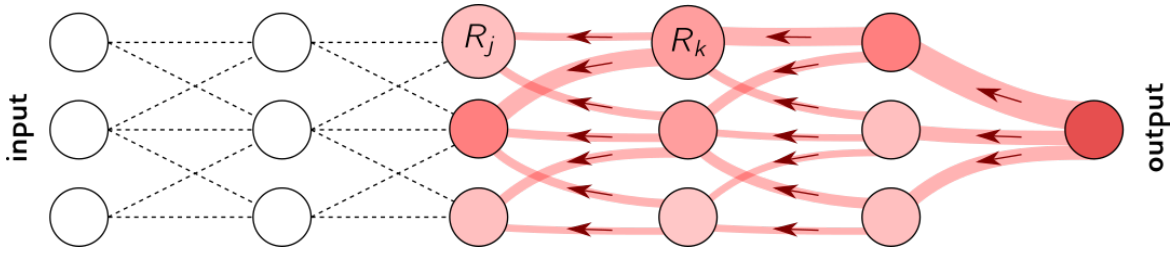


Fig. 6.7 Visualization of the LRP backpropagation procedure. The thickness of the lines shows the share of each neuron in the relevance in relation to the neurons of the next layer [87].

The standard backpropagation rule is based on the assumption that the amount of relevance should be preserved in each layer. Since we know the relevance of the neurons of the output layer, we can propagate the relevance layer by layer back to the input layer. The relevance $R_{i\leftarrow j}^{(l,l+1)}$ of a neuron i in layer l regarding a neuron j in the upper layer $(l+1)$ can be calculated the following way [12]:

$$R_{i\leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j} \cdot R_j^{(l+1)} \tag{6.2}$$

$R_j^{(l+1)}$ is the relevance of the neuron j of the layer $(l+1)$, $z_{ij} = a_i w_{ij}$ is the activation a_i multiplied with the weight w_{ij} of the connection between the neurons i and j and $z_j = \sum_{j=0}^{J} a_i w_{ij}$ is the sum over all connections from neuron i to the J neurons of layer $(l+1)$.

The total relevance of the neuron i can be calculated by summing up the relevance of all neurons of the layer $(l+1)$ regarding the neuron i [87]:

$$R_i^{(l,l+1)} = \sum_{j=0}^{J} \frac{z_{ij}}{z_j} \cdot R_j^{(l+1)} \tag{6.3}$$

Equation 6.4 shows the conservation of the relevance while propagating it from layer to layer [67]. I is the amount of neurons of layer l. The equation shows that the sum of relevances of the neurons of layer l regarding the neuron j of layer $(l+1)$ is equal to the relevance of neuron j. Therefore, relevance of that neuron is neither lost nor is any relevance added.

$$\sum_{i=0}^{I} R_{i\leftarrow j}^{(l,l+1)} = R_j^{(l+1)} \tag{6.4}$$

The rule explained in Equation 6.3 is called the *LRP-0* rule. It fulfills the relevance conservation requirement but has some drawbacks. Therefore, some alternative rules have been defined which are briefly explained in the following sub-sections.

The LRP-ε rule

One problem of the *LRP-0* rule is that $R_{i\leftarrow j}$ can take unbounded values for small values z_j. This can be prevented by introducing a stabilization factor ε. Equation 6.5 shows the resulting formula [67]. The stabilization factor leads to a better numerical stability by absorbing very small relevances. This of course also leads to a relaxation of the relevance conservation property [67].

$$R_{i\leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j + \varepsilon \cdot sign(z_j)} \cdot R_j^{(l+1)} \tag{6.5}$$

The LRP-$\alpha\beta$ rule

The *LRP-$\alpha\beta$* rule is specialized for NNs which use ReLU activation functions. It treats positive and negative relevance separately what allows us to chose the impact of positive and negative relevance. The equation is shown in Equation 6.6. Negative and positive contributions to the relevance are marked with $-$ and $+$ [67].

$$R_{i\leftarrow j}^{(l,l+1)} = \left(\alpha \frac{z_{ij}^{+}}{z_{j}^{+}} + \beta \frac{z_{ij}^{-}}{z_{j}^{-}} \right) \cdot R_{j}^{(l+1)} \tag{6.6}$$

Based on the different rules explained above, the LRP approach can be used to generate a visual explanation for the decision of a DNN which can be optimized according to the specific research question. Due to its properties the LRP approach is optimally suited to generate the visualization for the *Manufacturability Analysis* task described in Section 3.1. The two building blocks described above provide the basc functionality for our solution. In the next Section, we will describe how the raw 3D models have to be prepared to enable them to be processed by our solution.

6.4 Data Preprocessing

For being able to train the HCNN approach based on the gathered production data, several intermediate steps are necessary to create the data representation which is processible for the HCNN based on the 3D models which should be used as input data.

Figure 6.8 shows the steps which are necessary to create the input data for the HCNN. The process starts with the 3D models which are stored as STL or Wavefront 3D object (OBJ) file. As described in Section 4.4, the HCNN is a voxel-based approach which is able to process the 3D models with a high resolution of up to 512^3 voxel by using PSH for an optimized compression of the sparse voxel models. The PSH method is used to calculate a hash-table representation of the voxel models. The STL respectively OBJ model must first be converted into the hash-table format respectively batches of hash-table files which can then be processed by the HCNN approach. Several intermediate steps are necessary.

Prior to the creation of the voxel models, a point cloud representation has to be created based on the original 3D models. For that purpose, we use a modified version of the ray-tracing-based algorithm *Virtual Scanner* [143]. The *Virtual Scanner* is used for the creation of point clouds for e.g. 3D object recognition tasks using point-cloud-based DNN approaches. Via virtual ray-tracing, point cloud representations of 3D meshes can be generated. The points are calculated by casting a ray from a specific view point and calculate the intersection

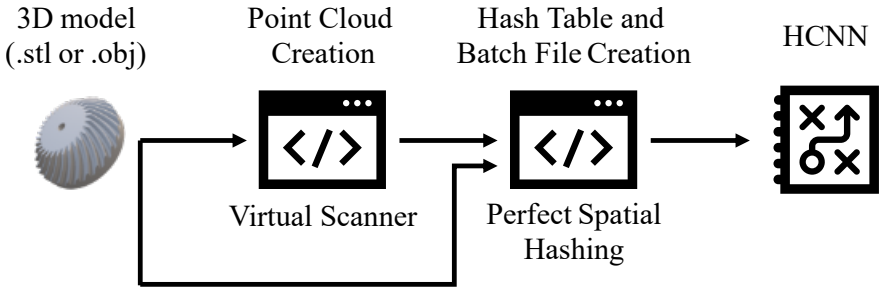


Fig. 6.8 Intermediate steps from the 3D model to the HCNN input batch-file.

of the ray with the 3D mesh [34]. For classical 3D object recognition tasks, only the external, visible surface of a 3D model is of interest. Therefore, the *Virtual Scanner* is implemented in such a way that only the intersection points with the outer surfaces are taken into account when creating the point cloud. For our *Manufacturability Analysis* application, however, we also need the geometric information of internal structures such as cooling channels.

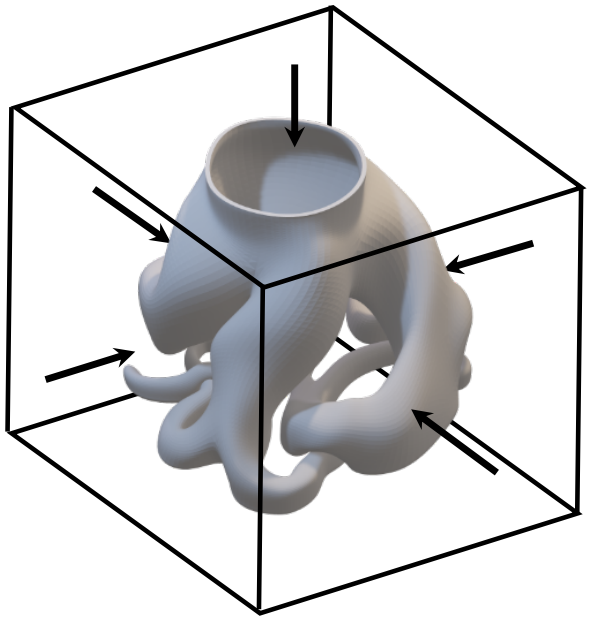


Fig. 6.9 Positioning of the six ray-arrays.

Therefore, our modified version of the *Virtual Scanner* is not only using the first intersection of the ray with the 3D mesh but also the following intersections. We use six view points from which our rays are cast on the model (See Figure 6.9). At each of the positions, a two dimensional ray-array consisting of 1024^2 rays is used to generate a detailed point cloud from each view point. The final high resolution point cloud is generated by merging the point clouds corresponding to the different view points.

Based on the point cloud, the actual voxel model of a 3D model can be generated. We are using an implementation provided by the authors of the HCNN approach [119]. For each voxel, the algorithm checks whether at least one point of the point cloud is located within the voxel or not. If this is the case, the voxel is marked as part of the surface of the 3D model. Thus, the voxel model represents a surface model of the 3D model which makes it very sparse. For the filling of the voxels, there are two variants. Either, they are simply stored in binary format, marking whether a voxel is part of the surface or not or, additionally the surface normal vectors are stored whereby additional information can be used. For the second variant, the surface normal vectors are extracted from the original 3D model and stored using the normalized length of their X-, Y- and Z-component in three separate channels. The training instances therefore consist exclusively of the geometric data, stored as hash-table files and the corresponding label manufacturable or non-manufacturable. For the non-manufacturable 3D models, no information is included regarding the exact non-manufacturable partial geometries within the 3D model. We have deliberately omitted this information in order to verify whether our solution can be trained using solely the labels. For a later use of real production data, the effort for labeling the data could be reduced significantly and thus the labeling process could be integrated more easily into the daily processes.

Subsequently to the generation of the voxel models, the hash representation can be generated, as explained in Section 4.4. The result is a hash-table-file (.hst-file) for each voxel model. In addition, for being able to process multiple 3D models in parallel, batch files (.HSLB-files) are created which can be loaded and processed by the HCNN approach.

6.4.1 Data Augmentation

To create a performant system that robustly learns the features to be learned, it is necessary to appropriately prepare the information of the raw 3D models using data augmentation techniques. In the real applications, arbitrary 3D models are uploaded to the web platform by customers and subsequently analyzed. One degree of freedom that exists in this process is the orientation of the 3D models. Regardless of the orientation in which a 3D model is uploaded, the evaluation regarding the manufacturability of the 3D model generated by our solution should always remain consistent. Since the HCNN model is not completely rotation-invariant, different orientations of the 3D models can lead to different decisions of the HCNN model with respect to the manufacturability of a 3D model although the manufacturability criteria we are dealing with are actually rotation-invariant. Therefore, it is necessary to eliminate this rotation variance with data augmentation methods as far as possible.

To counteract the rotation-variance, we want to provide as much information as possible to the HCNN model. Therefore, we generate the voxel representation and the related hst-file in different orientations for each 3D model. We decided to generate 14 different orientations for each 3D model in the voxelization process and accordingly 14 different hst-files for each 3D model. The different orientations for the 3D models are generated by creating an axis-aligned unit cube around the center of the 3D model and subsequently rotating it in a way that the vector formerly pointing in negative Z-direction is subsequently pointing in the direction of the normal vectors of the six sides of the cubes respectively to the 8 corners of it, one after the other. With this procedure, we generate 14 different hst-files for each 3D model. As a result, we provide different variants for each 3D model to the HCNN model during the training process, enabling it to obtain additional information about each 3D model.

6.5 Decision Calculation

In the training process as well as in the inference phase, the HCNN model is calculating a decision if a 3D model is manufacturable or not for each hst-file. Since we are creating the hst-files using 14 different orientations for each 3D model, as described in the previous Section 6.4, we created two different variants for the calculation of the final decision if a 3D model is rated as manufacturable or not. For the first variant, we have considered each input instance, i.e. hst-file generated from a particular point of view, individually. For the second variant, we considered each of the individual scores generated by the HCNN for the 14 hst-files per 3D model as a joint instance in the inference phase. For each of the 14 hst-files corresponding to a 3D model, an independent decision if the 3D model is manufacturable or not is still calculated by the HCNN, but these are subsequently used to determine a joint decision for all 14 hst-files. We rate a 3D model as non-manufacturable as soon as the HCNN has rated at least one of the 14 instances as non-manufacturable.

This is motivated by the fact that for non-manufacturable components, an explicit geometric feature has to be recognized, whereas for manufacturable components the non-existence of this feature is decisive. Since the decisive features are not necessarily recognizable for the HCNN model in all orientations of a 3D model, it is sufficient if they are recognized for one of the orientations. We will analyze the effects of these two variants as part of the evaluation in Chapter 8.

Our solution described in this chapter can be used for the *Manufacturability Analysis* in the process chain of AM service providers. It can be trained based using production data categorized and labeled with regard to certain manufacturability criteria. After training it

can be used for an automated evaluation of 3D models which have been uploaded to the web platform of an AM service provider by customers. The two stage solution is able to classify 3D models into the classes manufacturable and non-manufacturable objects using the HCNN stage and subsequently generate visual feedback using the LRP method.

In Chapter 8, we will present our work on a concrete realization of our solution. We created a demonstration data set labeled with regard to the manufacturability criterion *minimum wall thickness* and trained our solution to enable it to evaluate that criterion. Subsequently, the evaluation of the performance of our solution with regard to the requirements $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$, which we defined in Section 3.1.3, is provided. Before that, however, we describe our solution for the *3D Component Recognition* for additively manufactured parts in Chapter 7.

Chapter 7

Solution: 3D Component Recognition

In Chapter 3, we have outlined that the current approach of manually assigning the physical components to the corresponding digital 3D models and therefore the correct customer order is extremely time consuming and expensive. Already at the current production volume, it is only conditionally feasible, with increasing production volumes, the recognition and assignment of the components will no longer be possible in the current manual form. Therefore, our goal is to design an automated solution. In Section 5.2, we provided an overview of existing solutions from the AM domain respectively solutions for related tasks in other domains and rated these solutions regarding the solution requirements $R_{1-cr} \mathrel{\widehat{=}} \textit{recognition rate}$, $R_{2-cr} \mathrel{\widehat{=}} \textit{robustness}$, $R_{3-cr} \mathrel{\widehat{=}} \textit{adaptability}$, $R_{4-cr} \mathrel{\widehat{=}} \textit{process time}$, $R_{5-cr} \mathrel{\widehat{=}} \textit{scalability}$ and $R_{6-cr} \mathrel{\widehat{=}} \textit{process costs}$, which we defined in Section 3.2.2. Based on that information, we decided that an DL-based solution is the best choice for our problem statement. Based on evaluations using benchmark data sets like the Princeton ModelNet data set [146], DL-based approaches have shown that they are currently achieving the state-of-the-art accuracy in object recognition tasks and should therefore be well suited with regard to the requirement $R_{1-cr} \mathrel{\widehat{=}} \textit{recognition rate}$. Another very important aspect is that its adaptive data-driven nature makes it best suited with regard to the requirement $R_{3-cr} \mathrel{\widehat{=}} \textit{adaptability}$. DL-based solutions are able to automatically adapt to the daily changing components to be recognized. Since DL systems require comparatively little computational effort in the inference phase, it can also be expected that the constraints with respect to the requirement $R_{4-cr} \mathrel{\widehat{=}} \textit{process time}$ can be met. Additionally, we figured out that an image-based approach should be preferred over 3D data-based approaches with regard to the requirement $R_{6-cr} \mathrel{\widehat{=}} \textit{process costs}$, since cameras are usually cheaper than 3D scanners. This property should also make the solution scalable relatively easy. With regard to the requirement $R_{2-cr} \mathrel{\widehat{=}} \textit{robustness}$, our experiments will have to show that the requirement can be fulfilled by our solution. In this chapter, we will describe our solution for the *3D Component Recognition* of additively manufactured parts.

At first, we will explain the basic architecture of our solution in Section 7.1. Subsequently, we go into detail for the different building blocks of our solution. In Section 7.2, we will describe the DL architecture, which we used as central element of our solution. In Section 7.3, the concept for the physical recognition station will be explained. Subsequently, in Section 7.4 we explain the necessary steps for the training data generation and in Section 7.6 the preprocessing of the inference data used for the actual recognition of the components is described.

7.1 Solution Architecture

In Section 5.2, we explained our decision to develop an image-based DL approach for the *3D Component Recognition* task. In this chapter, we will explain the developed system in detail. Our solution is using the digital information of the virtual 3D models for the recognition of the physically produced components. Therefore, the solution is based on a composition of software and hardware building blocks. For the development of a suitable solution, it is essential that the hardware and software components are optimally adapted to each other in order to be able to link the digital information with the physical circumstances. We first describe the basic process flow of our solution for *3D Component Recognition* and subsequently explain the different building blocks in detail.

In Section 3.2.2, we explained the requirements which possible solutions for the *3D Component Recognition* have to fulfill. Based on these requirements, we have chosen to use an image-based DL approach for our solution architecture. Regardless of the exact DL architecture, which we will discuss in Section 7.2, this choice results in a process flow which consists of the steps shown in Figure 7.1.

Fig. 7.1 The five main steps necessary for the *3D Component Recognition* from data set definition to inference.

As described in Section 3.2.2, we need a solution which is able to recognize the produced components build job per build job. This means that for each build job, a separate instance of the selected DL model is trained, which learns to recognize the associated components. By using pre-trained instances of the DL model, the training effort can be reduced. The preparation for the actual recognition of the produced components can start as soon as the build jobs are defined. At this point of the process chain, we have the information which

components will be produced in which build job and therefore know which components have to be recognized.

At this point, however, we only have data sets in the form of 3D models but not yet the images we need for the image-based DL approach. Thus, the next step is to generate the training data in the form of images. Since the components are only virtually and not physically available at this point, we can not use real images of the components but have to generate synthetic training data. As already described in Section 4.1.6, the synthetic training data must be adapted as precisely as possible to the physical inference data. Therefore, we developed a method for the generation of physically sound training data, which is a central element of our solution. The generation of the physically sound training data is fully automated. When a build job is fully configured, the necessary information for generating the training data can be accessed automatically and the training data is generated. We describe the exact procedure for the generation of synthetic training data in Section 7.4.

As soon as the training data is available, the training process itself can automatically be started, too. As shown in Figure 3.7 in Section 3.2.2, roundabout 21 hours are available between the completion of the build job preparation step and the *3D Component Recognition*. This time can be used for the training data generation and the training of the DL model itself.

When the training is finished, the DL model is able to recognize the components corresponding to a build job based on images of them. Therefore, images of the physical components have to be generated in the physical recognition station, which we are going to explain in Section 7.3. Based on the images, the DL model is able to recognize the components. As shown in Figure 3.7 in Section 3.2.2, for the recognition of all components which have been produced on a given day, roundabout two to four hours are available.

Based on this work flow, the DL model, which is able to recognize the associated components based on real images, can be trained for each build job fully automated in parallel to the real production of the components. In the following sections, we will explain the main building blocks which are used for our solution for the *3D Component Recognition* in detail. We will begin with the choice of the DL model, continue with the corresponding hardware setup and subsequently explain the training data generation method.

7.2 DL model: RotationNet

In this section, we will explain the basic choice of the recognition system which we are going to use and explain the system in detail. We have identified that an image-based DL system is the most suitable with respect to the defined criteria $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, $R_{2-cr} \mathrel{\widehat{=}}$

robustness, $R_{3-cr} \mathrel{\widehat{=}}$ *adaptability*, $R_{4-cr} \mathrel{\widehat{=}}$ *process time*, $R_{5-cr} \mathrel{\widehat{=}}$ *scalability* and $R_{6-cr} \mathrel{\widehat{=}}$ *process costs*, but we have not yet described the concrete system. As explained in Section 4.4, in the last years, several approaches have been developed for 3D object recognition tasks. Since nearly all of them are evaluated based on the Princeton ModelNet recognition challenge [146], we decided to use the approach with the best recognition rate regarding this challenge. The MVCNN-based approach RotationNet [56] achieves the current state-of-the-art accuracies with 97.37% for the ModelNet40 data set including 40 different object classes and 98.46% for the ModelNet10 data set including 10 different object classes. Therefore, we have chosen to use RotationNet as central element for our *3D Component Recognition* solution.

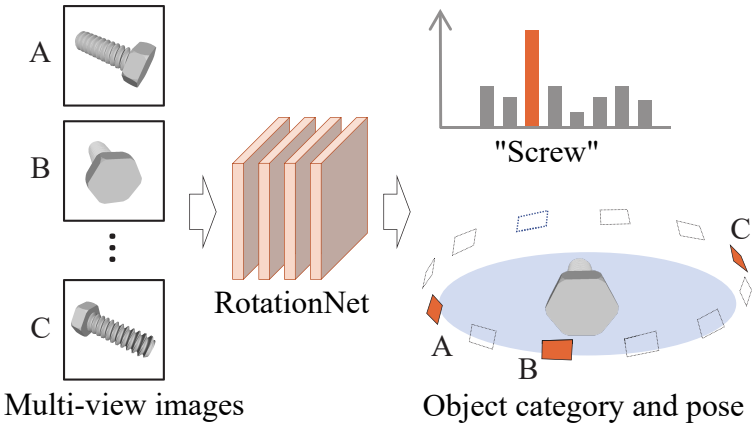


Fig. 7.2 Basic structure of RotationNet (Adapted from [56]).

RotationNet is only one of several approaches developed in the last years based on the first multi-view-based approach MVCNN [126]. Since its specific architecture enables it to achieve the state-of-the-art accuracy for the Princeton ModelNet classification challenge, we will explain the architecture in detail in this section.

The basic idea of RotationNet is to predict the category to which an object belongs as well as its pose based on e.g. 12 fixed camera perspectives (See Figure 7.2). Therefore, the viewpoints of the individual training images are treated as latent variables. That means that only the sequence of the viewpoints is known but not the exact positioning in relation to the object to be recognized. This unsupervised learning of the object pose allows an optimization of the calculation of class probabilities. Another aspect, which is especially important for practical applications, is that less physical cameras can be used in the inference phase than, for example, the 12 virtual cameras used in the training phase. That means that more virtual cameras can be used for the training data generation, without having the necessity of using the same amount of cameras in the inference phase.

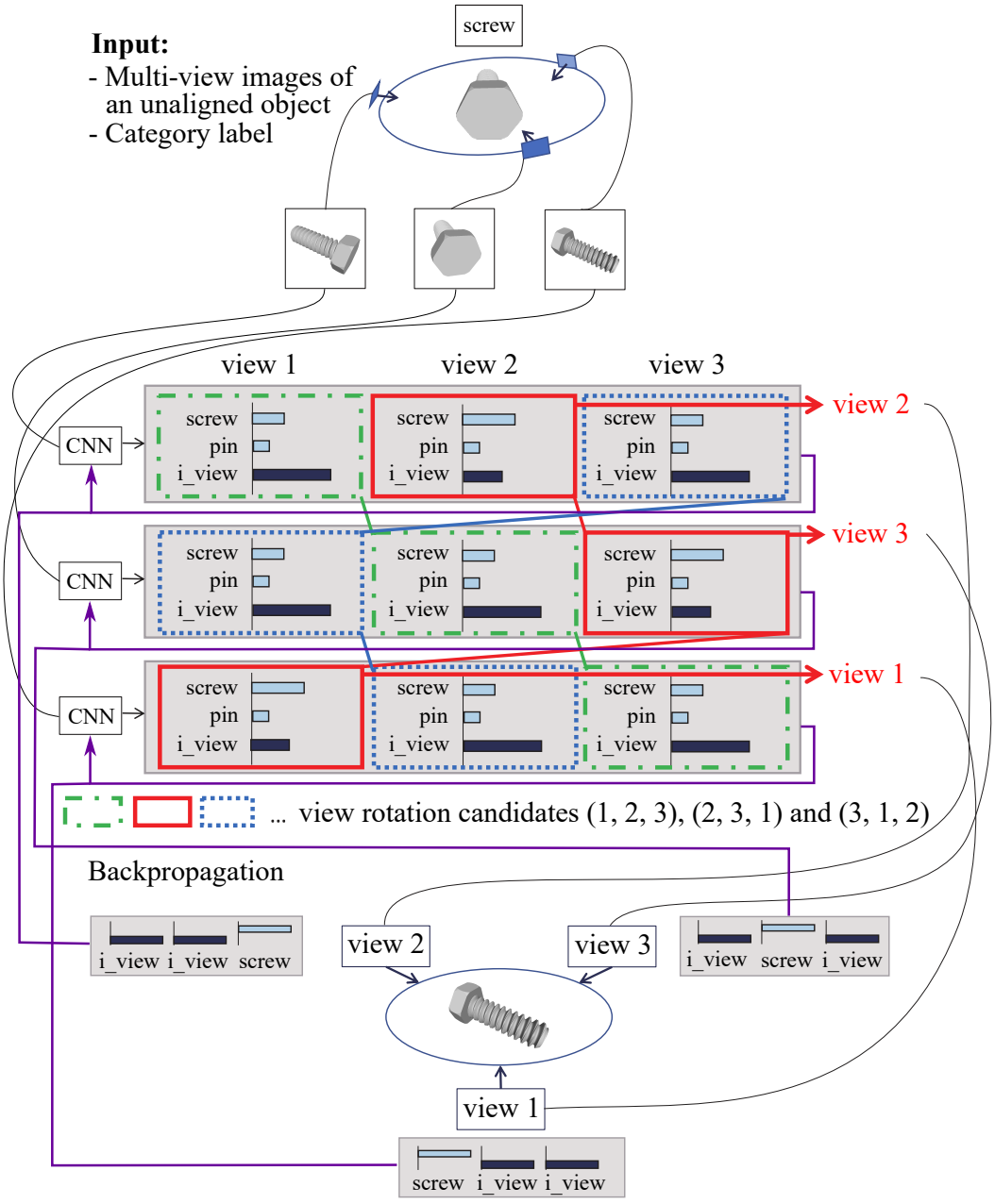


Fig. 7.3 Classification example using a system with $M = 3$ cameras (Adapted from [56]).

For the training process of a RotationNet model with M viewpoints and N different object categories, the training data is defined as follows: Each training instance $\{x_i\}_{i=1}^{M}$ includes M different images and the corresponding category label $y \in \{1,...,N\}$. Additionally, a viewpoint variable $v_i \in \{1,...,M\}$ is added to each training image x_i. These viewpoint are unknown and treated as latent variables. The output stage of the RotationNet architecture is designed as a concatenation of M softmax layers. Each of those layers predicts the category likelihood $P(\hat{y}_i|\mathbf{x}_i, v_i = j)$ for the corresponding input viewpoint x_i and the assumption that the viewpoint v_i is belonging to position j. The system is optimized for solving the optimization problem defined in Equation 7.1 [56].

$$\max_{R_i\{v_i\}_{i=1}^{M}} \prod_{i=1}^{M} P(\hat{y}_i = y|\mathbf{x}_i, v_i) \tag{7.1}$$

This equation is optimized in the training process by tuning the parameters of the RotationNet architecture and optimizing the latent variables $\{v_i\}_{i=1}^{M}$ in parallel for predicting the highest probability for y for a combined input instance $\{\mathbf{x}_i\}_{i=1}^{M}$ [56].

Figure 7.3 shows an example including $N = 2$ object categories and $M = 3$ cameras. In addition to the N object categories a third *incorrect view* category is added which represent views that can not be assigned to any target category. Therefore, for each of the M input images belonging to an instance, M output histograms, which contain $N + 1$ bins are calculated. For each of the histograms an other value for the latent viewpoint variable v_i is assumed and the probability distribution for the two categories and the *incorrect view* category are calculated. This allows to calculate which image probably belongs to which viewpoint.

In this example case, which includes three cameras, there are 3 possible options for the sequence of cameras: $(1,2,3)$, $(2,3,1)$ and $(3,1,2)$. Over all of these possibilities, the most likely category label $\hat{y}$ is calculated. In this example case, the highest combined accuracy for a valid class is generated when the viewpoint sequence $(2,3,1)$ is chosen [56].

The architecture described above, enables RotationNet to achieve state-of-the-art recognition rates in object recognition tasks. Especially the treatment of the viewpoints as latent variables makes it optimal suited for our problem statement. We will therefore use RotationNet as one of the main building blocks for our solution for the *3D Component Recognition* of additively manufactured components.

7.3 Physical Recognition Station

In the previous section, we described the RotationNet architecture, which is able to learn to recognize objects, using data instances of e.g. 12 images of an object. In order to use

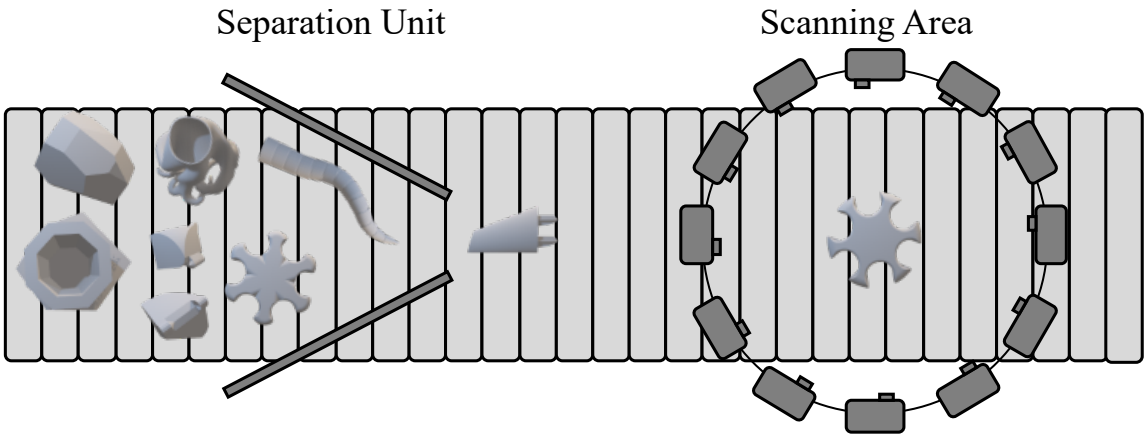


Fig. 7.4 Structure of the hardware of the recognition station shown from an aerial perspective. The components are transported on a conveyor belt. They are separated using a separation unit and afterwards scanned in the scanning area.

RotationNet for the recognition of additively manufactured components, we need to design the appropriate hardware to generate images of the components to be recognized. The recognition station has to be designed for the integration into a complete line automation of the entire process chain of AM service providers. For the design of the whole recognition system, not only the hardware of the recognition station itself is relevant, but also the hardware for the feeding of the components, since it influences their recognition by the method of moving the components in front of the camera. So far, we have only designed the recognition station virtually and have not yet built it physically. The design explained here describes a standard configuration and can still be adapted if necessary. Figure 7.4 shows the basic structure of the hardware shown from an aerial perspective. After the sand-blasting step, the components are placed on a conveyor belt. In order to be able to recognize the components one after the other, they are separated using a separation unit and subsequently conveyed further into the scanning area.

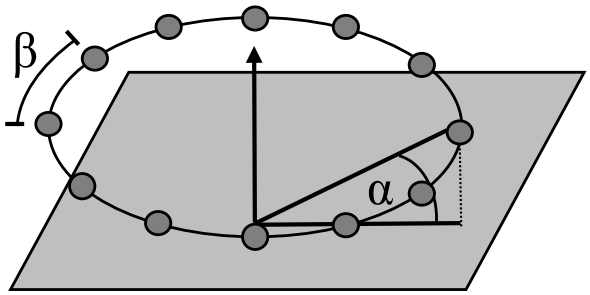


Fig. 7.5 Standard positioning of the 12 cameras of the recognition station.

The scanning are is designed based on the requirements of the RotationNet architecture. As explained in the previous section, the standard configuration of RotationNet consists of 12 camera perspectives. Therefore we have chosen 12 cameras as standard configuration for our design of the hardware station, too. However, this configuration can be adapted. The cameras are installed in a circle around the center of the conveyor belt. By default, the cameras are mounted at an angle $\alpha = 45°$ from the X-Y-plane and an angle $\beta = 30°$ around the Z-Axis between the different cameras (See Figure 7.5).

This hardware structure forms the basis for the recognition of the individual components. The cameras capture the components from 12 different angles and thus provide the information needed for the recognition of the components. The described hardware configuration is aligned to the architecture of RotationNet. In Section 4.1, we have described that the training and inference data for DL systems have to be aligned to each other as precisely as possible, to achieve a high performance regarding the task which is dealt with. Therefore, in addition, the further steps like the training data generation have to be adapted to the physical conditions of the hardware structure and the further restrictions of the overall process chain of AM service providers, as precise as possible. We will explain our developed solution for the training data generation in the following Section 7.4.

7.4 Training Data Generation

In order to enable the MVCNN architecture RotationNet to be able to recognize the components after production and sand-blasting on the basis of the camera images, it must first of all be trained to be able to recognize the components. Before the training can be started, we need a set of training data for each build job. Since we are using an MVCNN-based approach, we need multiple images of the components for the training process. However, as already mentioned in Section 7.1, when the build job preparation is completed and the training could in principle start, the components do not yet physically exist. Therefore, there are no images of the physical components that can be used for the training process. Instead of physical images, synthetic training data must be generated as described in Section 4.1.6.

In our case, synthetic training data can be generated by rendering virtual images based on the 3D models corresponding to the physical components, which later on have to be recognized. In Section 4.1.6, we explained that a precise adaption of the synthetic training data to the physical conditions in the inference phase is of decisive importance for being able to achieve a robust and efficient recognition system. Therefore, several intermediate steps are necessary to achieve an automated recognition of the physically produced components based on the information of the 3D models of the assembled build job.

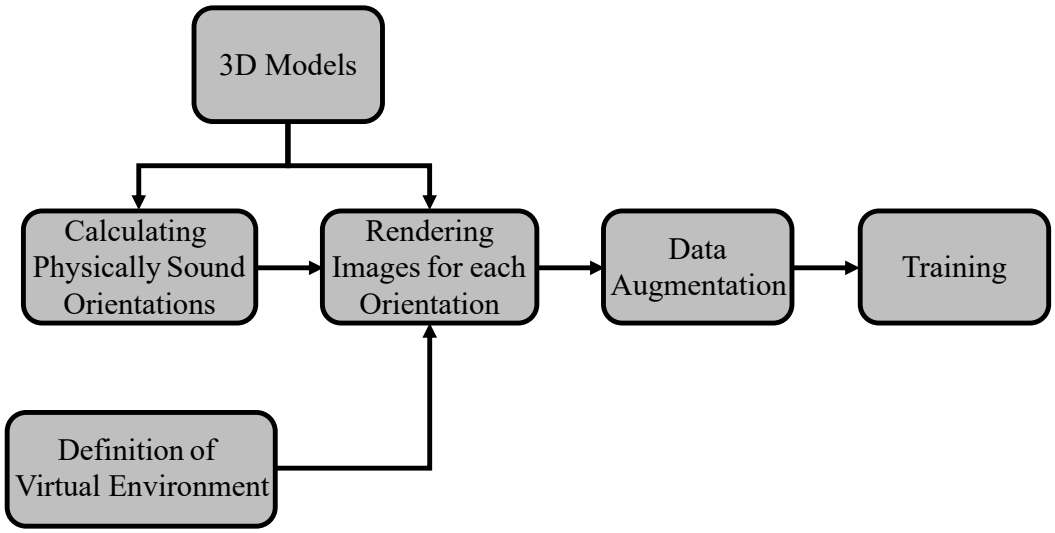


Fig. 7.6 The different steps for the creation of training data for an MVCNN-based recognition system based on a set of 3D models corresponding to a build job.

The different steps for generating training data for an MVCNN-based recognition system are shown in Figure 7.6. Before the rendering can be started two prior steps are necessary. Based on the 3D models, the orientation in which they are rendered later on has to be determined. This is done by calculating physically sound orientations for each 3D models. Using that procedure, the rendered images can be adapted to the physical behavior of the produced components on the conveyor belt. Additionally, the virtual environment, which forms the basis for the virtual rendering has to be defined. To achieve an optimal adaption between the synthetic training data and the physical images in the inference phase, the steps which we explained in Section 4.1.6 must be taken into account for the definition of the virtual environment. A high photo-realism is necessary and the whole physical scene including the environment and geometrical configuration with the positioning of the cameras and light sources have to be modeled for the synthetic training data generation. When both steps are finished, the rendering of the training images can start. To enhance the quality of the training data, data augmentation techniques as described in Section 4.1.7 are used subsequently. This allows the images to be even better adapted to the physical conditions in the recognition station, resulting in higher robustness of the whole recognition system. When those steps are completed, the actual training process can be started.

7.4.1 Virtual Environment

At first, we define the virtual environment using the open source 3D graphics software Blender [19]. Blender provides a Python Application-Programming-Interface (API) for setting up the

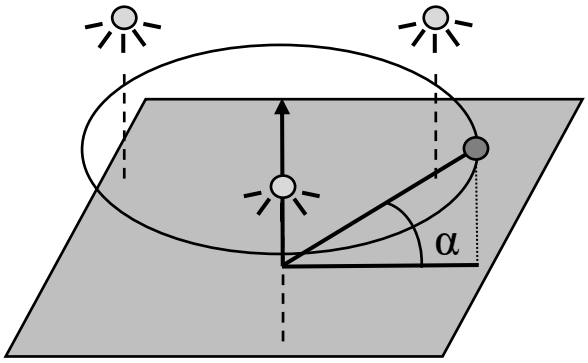


Fig. 7.7 Setup of the virtual scene including a camera and three point lamps.

whole scene including cameras, light sources, environment and objects including textures. A sketch of the virtual environment can be seen in Figure 7.7. We used three light sources in an elevated position around the center of the camera array. Since the physical recognition station is not existing yet, we defined this lighting conditions based on rendering tests. With this setup, we achieve a good illumination of the scene. The contours of the 3D models are highlighted without unwanted reflections or mirroring on the surfaces. For the background, we have chosen to use a simple black plane. Components produced via SLS are white. With the black background, we can increase the contrast between the objects and the background and therefore support the recognition process.

The cameras are set up according to the hardware setup defined in Section 7.3. Instead of adding 12 cameras to the virtual scene, we only use one camera which is moved to 12 different positions. For the experiments, which we will describe in Chapter 9, we used two different variants v_1 and v_2, which are shown in Figure 7.8. The first variant v_1 is using a virtual camera setup similar to the physical setup described in Section 7.3: The cameras are positioned in $\beta = 30°$ steps around the Z-axis with a vertical angle of $\alpha = 45°$. For the second variant v_2, the cameras are placed in an alternating way using a vertical angle of $\alpha_1 = 35°$ respectively $\alpha_2 = 55°$ degree to the horizontal plane instead of a fixed angle of 45 degree. Thus, on two different orbits around the Z-axis, 6 virtual cameras are used in 60 degree intervals at each orbit . The positioning of the cameras of the upper orbit is offset by 30 degrees to the cameras of the lower orbit. In this way, inaccuracies in the generation of the physical images are already simulated during training.

Using the described virtual environment, we are rendering the objects from the 12 camera positions. We use Phong Shading provided by Blender Phong [73] to achieve photo-realistic rendering of the 3D models.

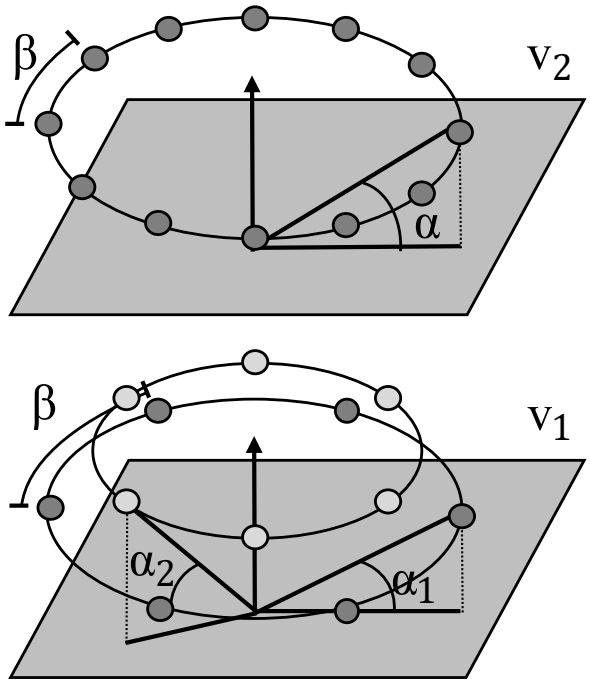


Fig. 7.8 The two variants for the camera settings for the training data generation. The upper setting is the standard variant.

7.4.2 Physically Sound Orientations

As shown in Figure 7.6, we second step which is necessary prior to the actual rendering of the training images is the calculation of physically sound orientations. For object recognition tasks, CNNs are usually trained with a large number of training instances for each object to be recognized. For image-based approaches, the instances of each object are represented by images from different perspectives. In our case, a training instance is not represented by a single image, but by the group of 12 virtual images of a 3D model created from the 12 camera perspectives. To create multiple instances for each 3D model, we need to view and render the 3D model from different perspectives. For this purpose, the 3D models must be rotated virtually into different orientations. Since, as already described, the training data must be optimally adapted to the physical conditions, we have developed an approach to create physically sound orientations which are used for an optimized rendering of the training data [95].

Physically sound means that we want to adapt the virtual training data to the physical conditions. We want to determine in which orientation the physical objects will probably be oriented in front of the camera array. If we calculate the most probable orientations of a physical object, we can generate the training data using the same orientations and thus adapt the virtual training data to the physical inference data. Instead of using randomly

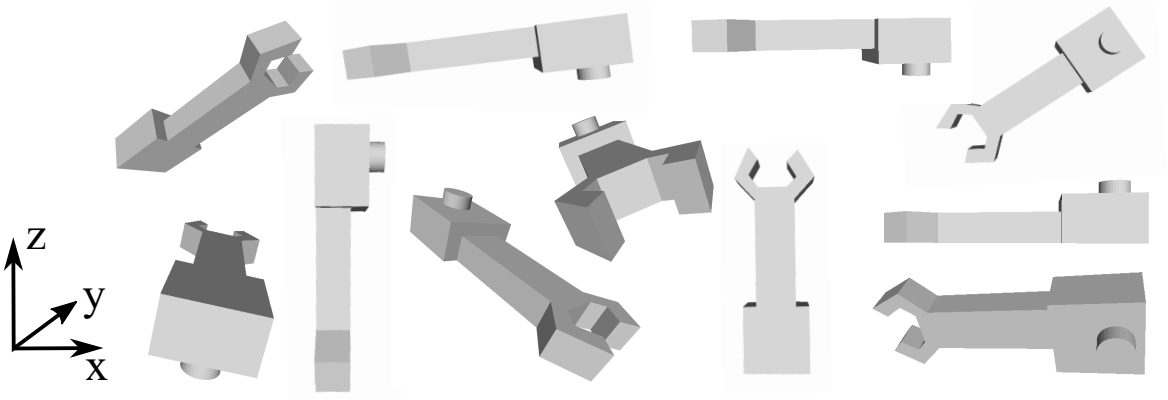


Fig. 7.9 An example object in eleven different orientations shown from a frontal view point.

chose orientations for the rendering, this procedure leads to a training data set for each component, which only includes images from viewpoints that can possibly occur in the physical recognition station. As we will show later in our evaluation, this adjustment has a high impact on the recognition rate of our recognition system.

Our goal is to determine which orientation an object is likely to be oriented in, in the recognition station based on the geometric properties of the corresponding 3D model. The process of calculating these physically sound orientations is divided into the following steps. To illustrate the effects of our approach, we show an example 3D model in different orientations (See Figure 7.9) and the effects of the individual steps of our approach on possible orientations. Figure 7.9 shows the 3D model from a frontal viewpoint in eleven different randomly chosen orientations. We will explain our approach using the following Figures 7.11, 7.13 and 7.14.

Convex Hull

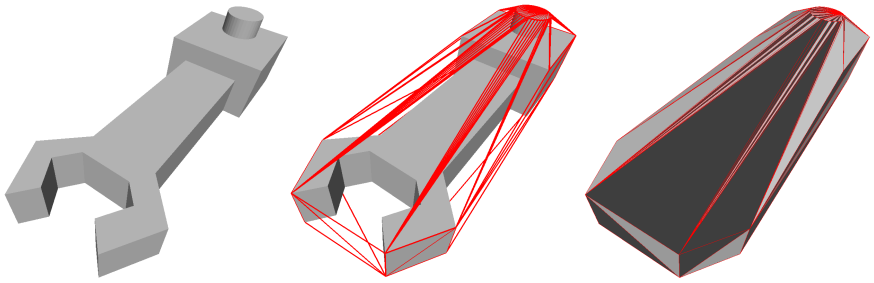


Fig. 7.10 An example object (left), the edges of the convex hull (center), and the facets created by adjacent points of the convex hull (right).

In the physical recognition station, the components will lie on a flat surface. Therefore, we first determine all orientations in which a component could in principle touch a flat surface, with more than a single point. For this calculation, we use the geometric property of the convex hull. An example is shown in Figure 7.10. The convex hull of a 3D object is described by the smallest convex set of points that contains the initial object (Figure 7.10 center). The facets that can be formed by connecting adjacent points of the convex hull represent all facets, with which the component can touch a plane surface (Figure 7.10 right).

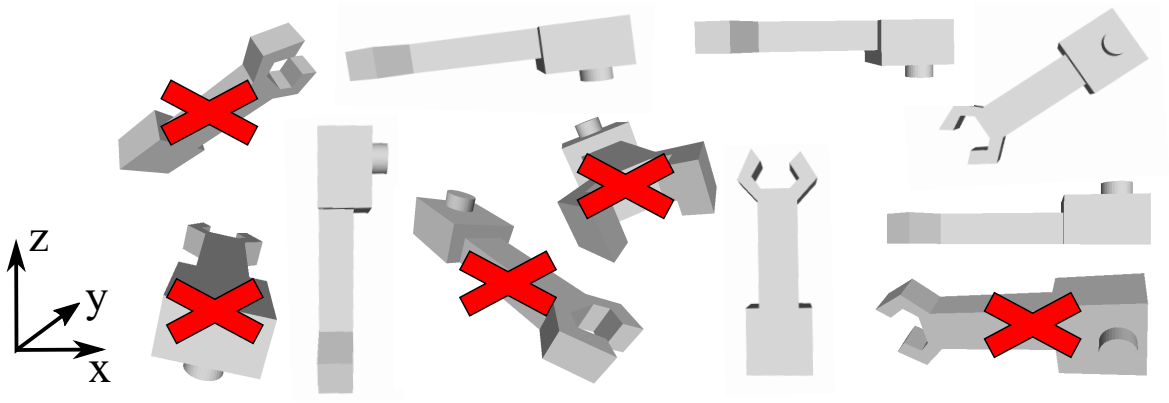


Fig. 7.11 Five of the eleven orientations can be excluded since for non of the facets of the convex hull the normal vector is pointing in negative Z-direction.

By using this property, the number of possible orientations in which a component can be lying on a horizontal plane can be limited from an arbitrary number to a much smaller number in most cases. Only if the normal vector of one of the facets of the convex hull points in negative Z-direction, the component can in principle lie on a horizontal plane. Figure 7.11 shows the reduction of possible orientations for the example 3D model. Five orientations can be excluded since non of the facets of the convex hull is parallel to the horizontal plane while its normal vector is pointing in negative Z-direction. This means that the associated component would not be lying plane on the horizontal surface in these orientations.

Object Stability

Using the properties of the convex hull, we were able to determine all orientations in which a facet of the convex hull is parallel to the horizontal plane and the corresponding normal vector points downwards. However, this does not mean that these orientations are physically stable. For the verification if a component is physically stable in a certain orientation, we are using the properties of the Center of Mass (CoM) of the component. The CoM of a component is the point at which the distribution of weight is equal in each direction.

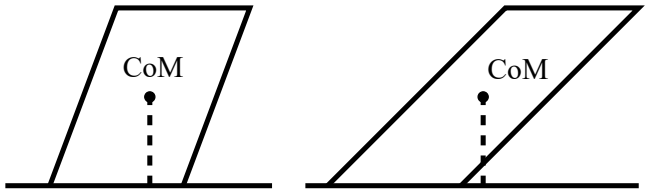


Fig. 7.12 2D example for the stability verification using the CoM. The left part of the figure shows a stable and the right part an unstable object.

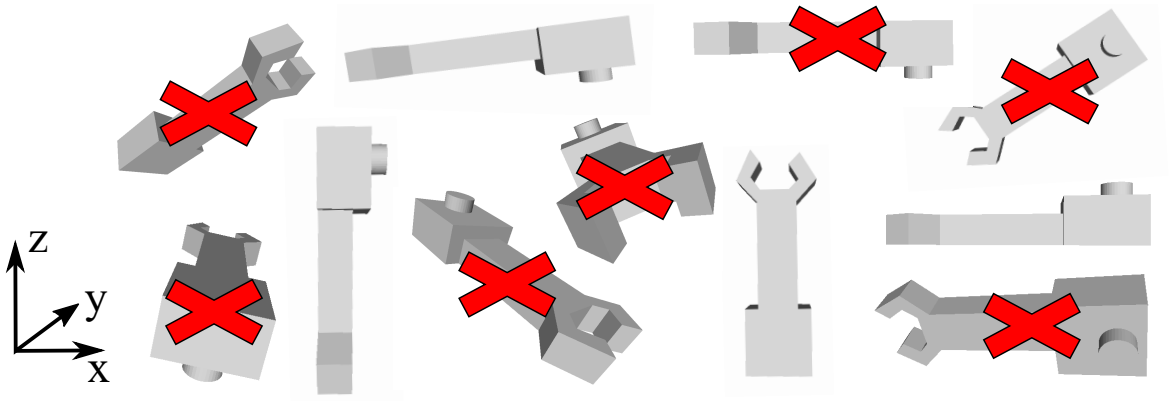


Fig. 7.13 Two additional orientations can be excluded using the stability verification based on the CoM.

A component is in a stable position if the projection of the x-y-position of its CoM is inside the convex hull of all points which contact the surface. A 2D example for the stability verification is shown in Figure 7.12. Based on this property, additional orientations can be excluded. Figure 7.13 shows the remaining possible orientations for the example component. Two additional orientations can be excluded.

Probability of Occurrence

All orientations which are remaining after the prior steps are physically stable and therefore represent possible orientations in which a component can in principle be located in front of the camera array. Nevertheless, further assumptions can be made about whether all calculated orientations will actually occur in the physical recognition station or not. Some of the orientations are much less likely than others. Components in orientations with a high CoM tend to tip over more easily than components in orientations with a low CoM. For example, theoretically a pencil could stand on its tip but in reality this is difficult to achieve. While being transported on the conveyor belt, the components will most likely move into an orientation with a low CoM. Therefore, in this step, we exclude all orientations with a very high CoM compared to the lowest possible CoM of an object. The exact threshold for the

exclusion will be defined in Chapter 9. Figure 7.14 shows the remaining orientations of the example component after this last step.

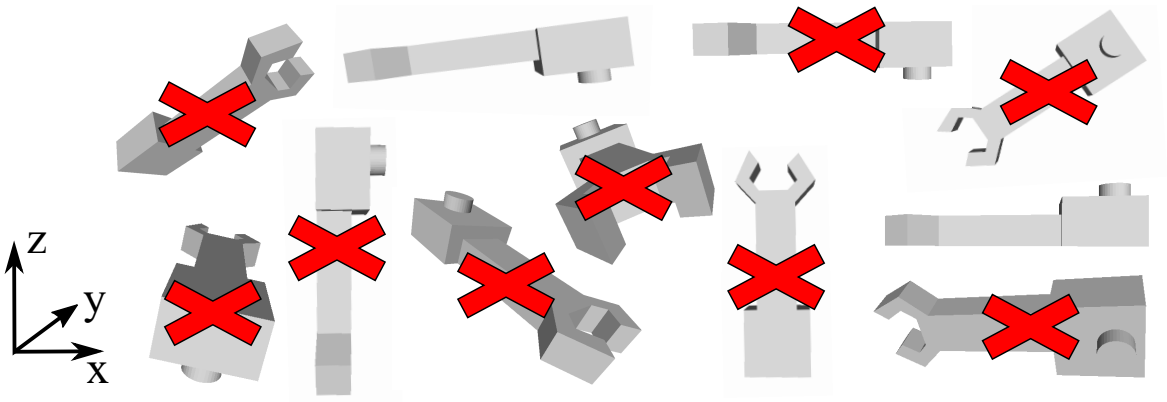


Fig. 7.14 Two additional orientations are excluded since they have a high CoM.

Using the steps described above, we are able to calculate the orientations in which the physical component will most likely be oriented in front of the camera array, based on the properties of the virtual 3D model. With this information and the virtual environment we have defined, the actual training data can subsequently be generated.

7.4.3 Image Rendering

The training data instances in form of batches of 12 images are generated via rendering of virtual images using the virtual environment and the physically sound orientations of the 3D models. The images are rendered using a Phong Shading implementation provided by Blender Phong [73]. We create images with a resolution of 250 by 250 pixels. Based on first experiment series, it has been shown that the best results can be achieved if the 3D models are normalized to a uniform size for rendering. The 3D models are therefore normalized using their maximum dimensions, so that they fit into a unit sphere. By positioning the camera on a sphere with radius two, all 3D models are rendered from the same distance. Figure 7.15 and 7.16 show the same example training instance rendered using the two camera array variants v_1 (Figure 7.15) and v_2 (Figure 7.16).

To enable an optimal training process, we want to generate balanced training data, i.e. an equal amount of training instances per 3D model. Unbalanced training data can lead to a deterioration of the recognition rate of our recognition system [82]. With the described method for calculating physically sound orientations, a varying number of physically sound orientations is calculated for each 3D model depending on its geometry. We use these as basic orientations. To be able to create the same number of training instances for each 3D

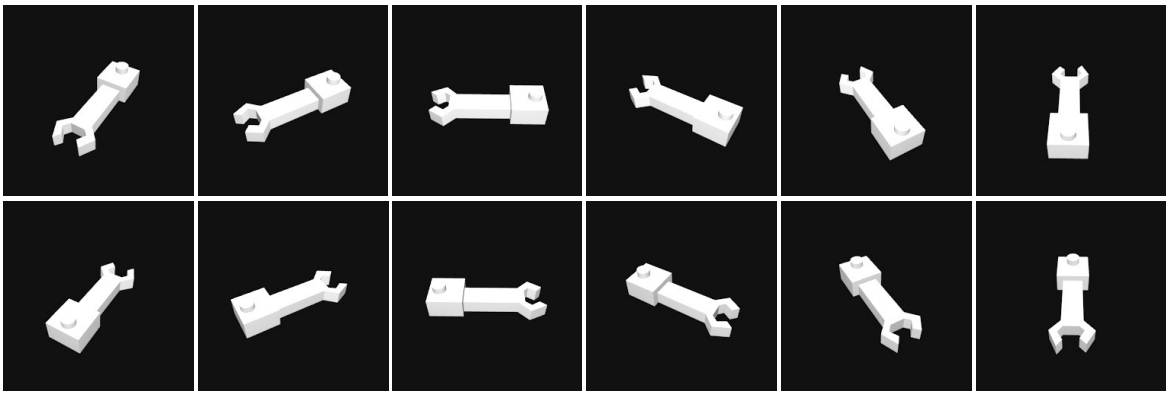


Fig. 7.15 Example of a training instance consisting of 12 images rendered using the camera array variant v_1.

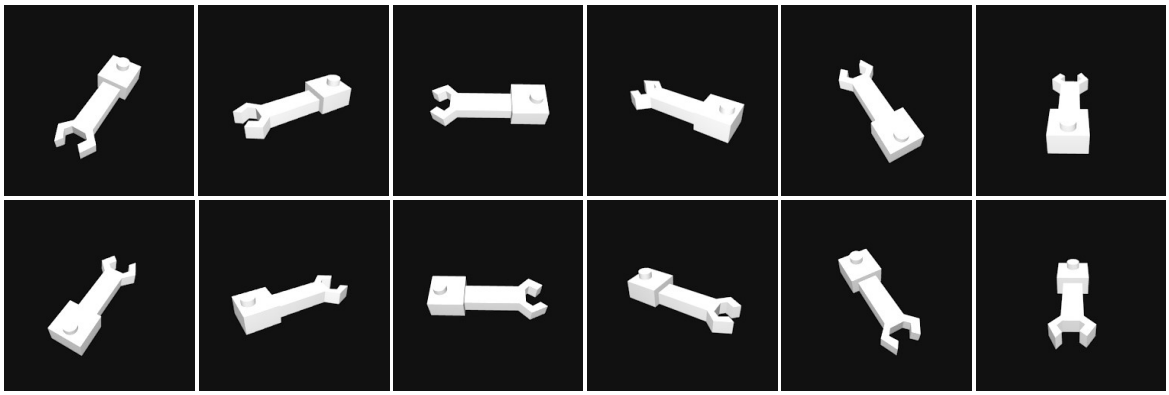


Fig. 7.16 Example of a training instance consisting of 12 images rendered using the camera array variant v_2.

model, despite the varying numbers of possible basic orientations for the individual 3D models, we calculate additional stable orientations from these basic orientations.

In Section 7.4.2, we defined that a component is in a stable position if one of the facets of the convex hull is parallel to the horizontal plane and the corresponding normal vector is pointing in negative Z-direction. We can determine further valid orientations by rotating the 3D model around the Z-axis from a stable base orientation without violating the defined conditions. This allows us to calculate an arbitrary number of further stable orientations based on the basic orientations.

Independent of the number of base orientations M which we calculated for a 3D model, we can create the same number of N training instances for each 3D model by rotations around the Z-axis. For this purpose, we create further N/M stable orientations per base orientation for each 3D model and use these for the rendering process.

The single instances should be disjunct to each other. In order to avoid that the rotation creates identical instances, we have to determine for how many degrees the 3D models have to be rotated around the Z-axis. As defined in Section 7.4.1, the 12 virtual cameras are positioned in 30-degree steps around the Z-axis. Therefore, to avoid identical instances, we have to select rotation angles smaller than 30 degrees. Therefore, we rotate the 3D models $(N/M) - 1$ times for $30/(N/M)$ degree. If N/M is not an integer, we round up. In that case, more than N instances are generated for a 3D model. For training, N of these instances are randomly chosen to guarantee that the training data is completely balanced.

This process allows us to determine the same number of training instances for each 3D model, regardless of the geometry of the 3D models and can thus generate balanced training data. The images rendered that way are not directly used for the training data. As explained in Section 4.1.7, data augmentation techniques can be used to enhance the quality of the training data. We are using data augmentation techniques for an additional adaption of the training data to the physical conditions in the recognition station.

7.4.4 Image Augmentation

In Section 4.1.7, we provided an overview about different techniques for augmenting images. The described transformations are divided into geometric and photometric transformations. We use a subset of those techniques for the adaption of the raw virtually rendered images to the physical conditions of the recognition station.

The geometric operator rotation has already been integrated into our solution through the steps previously performed for the generation of training data. By calculating physically sound orientations, we generate a variety of different orientations that are used for rendering the 3D models. Additionally, the architecture of RotationNet itself provides a form of rotation. Through its multi-view architecture 12 different view points are used for each object. By using the different camera array variants v_1 and v_2, we add another rotation. Thus, through these steps, rotations are performed that augment the training data to cover as many situations that occur in reality as possible. Experiments conducted at the beginning of the evaluation have shown that further geometric transformations such as flipping or cropping do not lead to a further improvement of the performance of our solution. In the experiments described in Chapter 9, we will show that especially the physically sound generation of the training data has a crucial influence on the performance.

Besides the geometric transformations also the photometric transformations have a major impact on the performance of our solution. When the inference images are generated in the physical recognition station, there may be differences in the brightness, sharpness, contrast or other properties of the images due to changes in the environment. Since these can have a

massive impact on the recognition rate of our solution, we have also integrated the occurring effects into the training images through photometric transformations.

When using photometric transformations, it is not possible to predict which transformations are best suited to optimize the training data for a problem. We have used different variants in our experiments and evaluated their influence on the results. In first experiments, we used the photometric transformations blurring, Random Brightness/Contrast Change (RBC), Contrast Limited Adaptive Histogram Equalization (CLAHE), Gaussian noise injection and composition thereof.

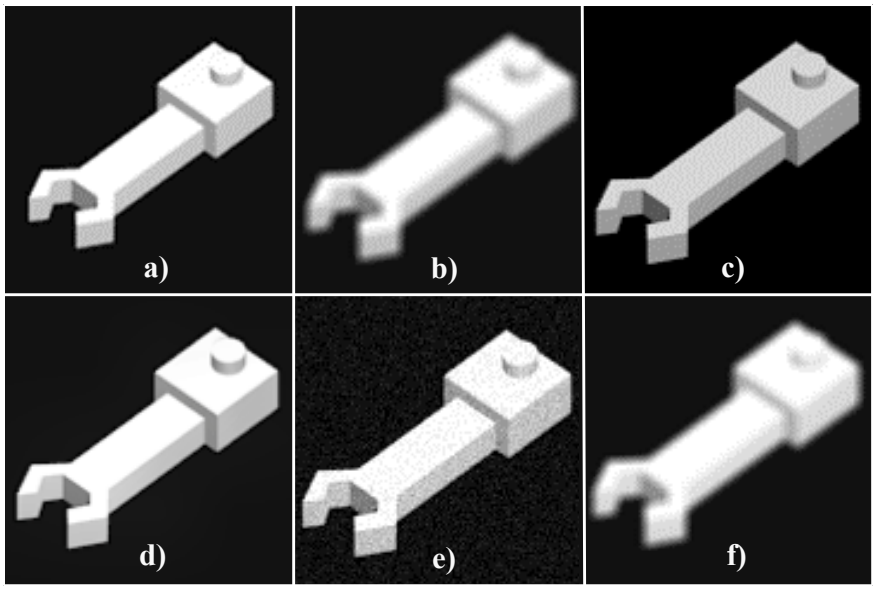


Fig. 7.17 Example for the five augmentation techniques which have been used. a): raw image, b): blurring, c): RBC, d): CLAHE, e): Gaussian noise, f): composition of augmentation techniques.

Examples for the different augmentation techniques are shown in Figure 7.17. Figure 7.17 a) shows the raw image, b) the image augmented using blurring, c) RBC, d) CLAHE, e) Gaussian noise and f) a composition of augmentation techniques where the composition randomly combines two augmentation techniques: one of Gaussian noise or RBC (probability of 50% each) and one of blurring or CLAHE (probability of 50% each). In the case of the composition and the RBC method, the parameterization of the filters is chosen randomly for each of the 12 images. So each image is augmented in a different way.

With these methods, we further augment the raw virtually rendered images to achieve optimally adapted training data. Based on these, the actual training process can start.

7.5 Training

The goal of the training process is to train a separate instance of RotationNet for each build job. Subsequent to the training, the trained instance can be used for the recognition of the physical components. Since the whole process from the generation of training to the recognition of the physical components has to be executed completely automated, also so the training process has to be started and especially stopped completely automated. The start of the training process can automatically be triggered as soon as the augmentation of the training data is completed. At that point, the whole set of training data for a build job is split into disjunct sets of training and validation data. During the training process, the validation data can be used to verify if the validation recognition rate is still growing. Using that validation steps, overfitting of the RotationNet model can be avoided. The RotationNet state which achieves the best validation accuracy is stored and used for the recognition of the physical components.

7.6 Inference Data Pre-processing

When a RotationNet instance is successfully trained, the state which achieves the best validation accuracy is available and the physical components are produced and prepared for the recognition, the actual recognition process can start. As explained in Section 7.1, the components are separated on the conveyor belt and moved into the scanning area. There the images of the physical objects are generated using the camera setting which we described in Section 7.3. Again, initial experiments have shown that augmenting the inference images also has an impact on the performance of our solution. Therefore, we applied different photometric augmentation techniques for them as well.

We used the augmentation techniques blurring, Gaussian noise, RBC, sharpening, a combination of sharpening and CLAHE and the composition of Gaussian noise, RBC , blurring and CLAHE as explained in Section 7.4.4. The techniques blurring, Gaussian noise and the composition of Gaussian noise, RBC , blurring and CLAHE lead to significantly lower recognition rates in the experiments. Therefore, we continued with the three techniques RBC, sharpening and the combination of sharpening and CLAHE shown in Figure 7.18 for the subsequent experiments.

Using the approach defined above, we are able to create the link between the information of the digital data process chain and the physical process chain for the automated recognition of additively manufactured components. We are using the MVCNN-based architecture RotationNet and train it based on physically sound synthetic training data to enable the system

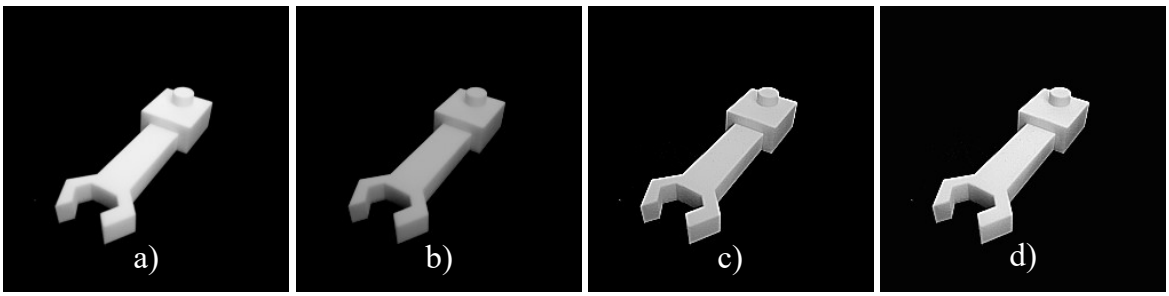


Fig. 7.18 Example for the physical evaluation images: a) raw image, b) image augmented using RBC, c) image augmented with a sharpening filter, d) image augmented using a sharpening filter and a CLAHE filter.

to recognize physical components. The entire process from the calculation of physically sound orientations over rendering, image augmentation and the training itself can be fully automated and integrated into the process chain of AM service providers. As a result, our solution is able to adapt to the changing customer orders on a daily basis in a fully automated manner and subsequently provide an automated recognition of the produced components. In Chapter 9, we will explain how the solution described above is realized using a concrete demonstration setup. We perform all the necessary steps described above and evaluate the performance of our solution with regard to the requirements defined in Section 3.2.2. Previously, in the next chapter, we present the realization and evaluation for the process step *Manufacturability Analysis*.

Part V

Realization, Evaluation and Conclusion

Chapter 8

Realization and Evaluation: Manufacturability Analysis

In this chapter, the realization and evaluation of our developed solution for the *Manufacturability Analysis* in the process chain of AM service providers, which we explained in Chapter 6, will be presented. Since the solution was developed for being used in the process chain of AM service providers, the experiments conducted for the evaluation are designed to represent the real processes as precise as possible. The developed solution should be able to learn to evaluate the manufacturability of unknown 3D models based on production data in the form of 3D models labeled in terms of their manufacturability. In addition to simply deciding whether a component is manufacturable or not, visual feedback regarding critical geometries must be generated. In this chapter, we describe the concrete realization of our developed solution. This means that we apply the developed solution, including all solution steps described in Section 6, to an appropriately generated data set. We will describe the realization including the data set definition and the experimental setup in Section 8.1. Subsequently, in Section 8.2, the evaluation of our solution with regard to the defined requirements $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} feedback$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ which we described in Section 3.1.3 will be described. In Section 8.3, we will discuss the presented experimental results and subsequently conclude the chapter by providing a summary in Section 8.4.

8.1 Realization

The basic idea of our approach is that instead of manually generating and using design rules which often only approximate manufacturability constraints, existing constraints can directly

be learned based on production data which is generated in the daily production of AM service providers. In contrast to the design rules described in Section 3.1.1, the information generated during the daily production of AM service providers does not contain approximations of the actual constraints but contains information about the actual occurring physical constraints in the real production. It is our goal to extract this information with the help of the developed data-driven solution and to use it for the evaluation of new components.

At the current state, however, we do not have sufficient production data and, in particular, information on production errors to train our system on the basis of this data. In addition, to the best of our knowledge, currently no benchmark data set for the *Manufacturability Analysis* for AM exists. To be able to evaluate our system nonetheless, we have decided to synthetically generate a labeled data set on our own. With the help of the synthetic data, we can carry out experiments that allow us to make a statement about the performance of our solution for the *Manufacturability Analysis* for AM. If our solution is able to learn a manufacturability criterion based on the synthetic data, the system should also be usable for being trained to learn further criteria using real production data. For this purpose, we first need a base data set and subsequently have to label it with respect to a criterion yet to be chosen.

8.1.1 Base Data Set

In order to be able to perform an evaluation that is as close to the real conditions, we want to generate a data set that reflects real production data as closely as possible. For this purpose, we used the Thingi10K database [152] as a base data set. The data set contains 10000 different 3D models from the AM domain, ranging from art objects to industrial components. Therefore, the data set is optimally suited to represent real conditions. Figure 8.1 shows some example objects from the Thingi10K data set.

8.1.2 Evaluation Criterion

As explained in Section 3.1.1, a diverse set of different criteria for manufacturability constraints exist. For being able to evaluate if our developed solution is able to learn geometric constraints based on labeled data, we need an example criterion with respect to which we label the Thingi10K data set. We have chosen to label the base data set with regard to the manufacturability criterion *minimum wall thickness*. The choice is based on two reasons: At first, this criterion is the most common cause of errors in real production. In addition, it is extremely time-consuming to manually label a large quantity of 3D models. The criterion

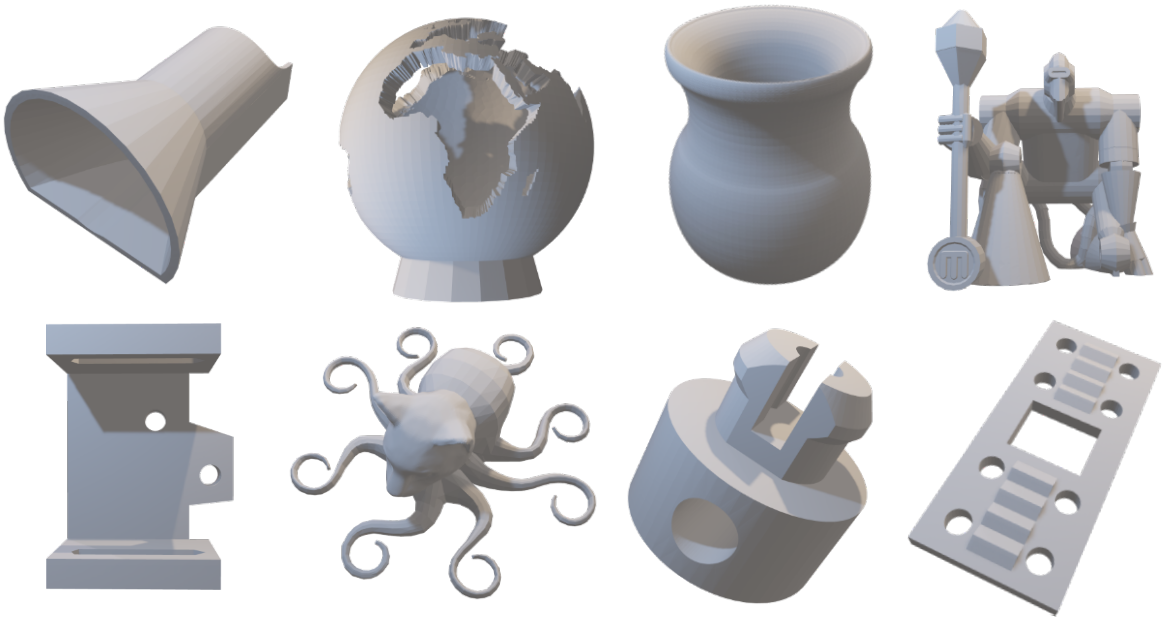


Fig. 8.1 Example 3D models from the Thingi10K data set [152].

of *minimum wall thickness* can be evaluated at least to a certain extent with the help of the software tools which we described in Section 2.2.

In Section 3.1.1, we have already explained the reasoning behind this restriction. For the PBF processes SLS and SLM, the *minimum wall thickness* which can be produced is bounded by the size of the laser spot and the layer thickness. Since these properties differ for different machines, the threshold values for the restriction also vary from machine to machine. In order to be able to evaluate the basic ability of our solution to learn this criterion based on labeled data, we have chosen 0.3 *mm* as threshold value for labeling our base data set. This value is lower than the standard values defined in the design rules of Adam et al. [2]. We have chosen this threshold value in such a way that the used Thingi10K base data set is approximately balanced divided into manufacturable and non-manufacturable components in order to enable proper training conditions for the training of our solution.

The choice of the threshold requires an additional constraint on the maximum extents of the 3D models that can be evaluated using our HCNN-based approach. As stated in Section 6.2, the HCNN is able to process 3D models with a resolution of up to 512^3 voxels using an NVidia GTX1080Ti GPU with 11GB memory and a batch size of 32 models for training. In order to enable the HCNN to detect the geometries with wall thicknesses below the threshold of 0.3 *mm*, the resolution must be adjusted accordingly. For being able to achieve both a high detail resolution and to be able to process 3D models that are as large as possible, we have decided to set the edge length of the voxels to 0.2 *mm*. Using this resolution, the

system is able to display relatively small features and process 3D models up to a size of $512 * 0.2\ mm = 102.4\ mm$. Using that constraint, we removed 5069 of the 10000 3D models from Thingi10K and used the remaining 4931 3D models for the following steps. In the following, we are going to describe the labeling of our data set with regard to the *minimum wall thickness* criterion.

8.1.3 Data Set Labeling

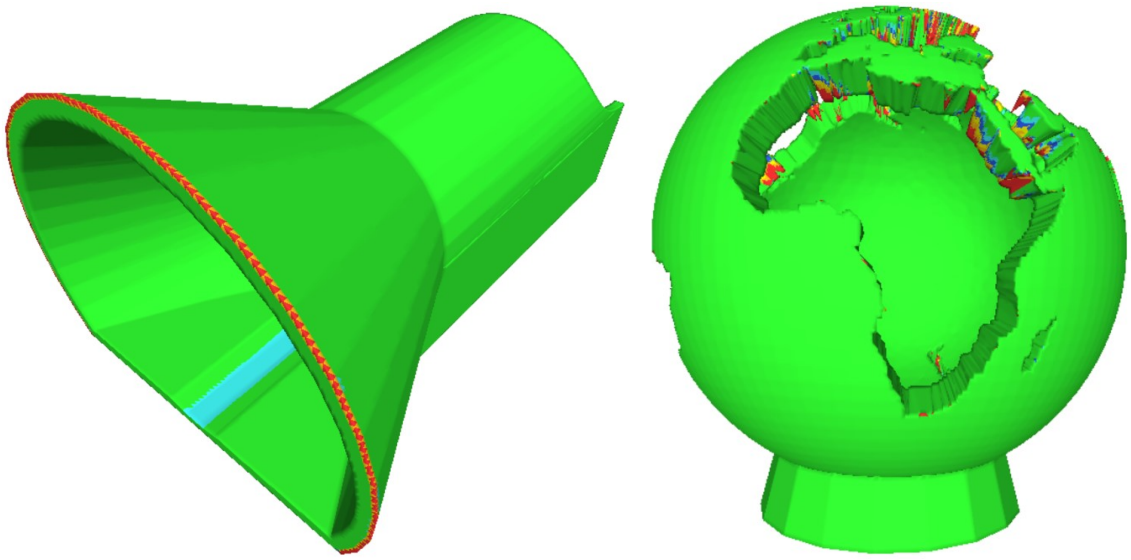


Fig. 8.2 Example 3D models from the Thingi10K data set [152].

The already selected data set must be labeled for our experiments with respect to the criterion *minimum wall thickness*. Since labeling of the 4931 exclusively manually is too time consuming, we decided to use the software tool available at the AM service provider for pre-labeling the data set. As already described in Section 3.1.3, the existing tools only provide a moderate evaluation accuracy. Therefore, we used the tool only for pre-labeling of the data and subsequently controlled the labels via a visual control.

Using this process, we can generate a high quality data set. Figure 8.2 shows an example of two 3D models which correctly have been labeled as being non-manufacturable with regard to thin walls. Areas of the geometry which are too thin are marked in red. Areas which are close to the threshold of 0.3 *mm* are marked from orange to blue up to a value of 0.6 *mm*.

In contrast to the correctly labeled 3D models shown in Figure 8.2, Figure 8.3 shows two example 3D models which have incorrectly been labeled as non-manufacturable. The incorrect labeling occurs due to failures in the mesh structure of the corresponding 3D

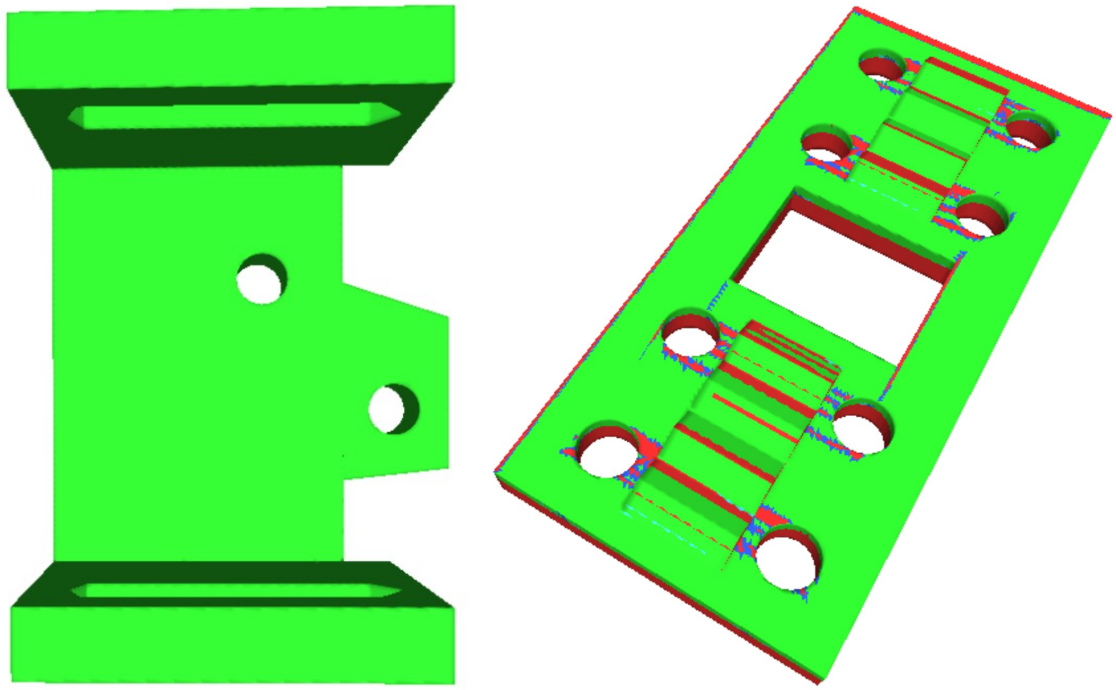


Fig. 8.3 Examples for incorrectly labeled 3D models.

models. When the 3D models are converted from any CAD type to STL or OBJ models, sometimes micro-errors occur in the mesh-structures. It can happen that the surface of a 3D model is partly represented by multiple mesh-layers. This leads to 3D models being rated as non-manufacturable although they are actually manufacturable.

For the example shown on the left of Figure 8.3 only a very small error in the mesh-structure leads to the wrong classification. For the 3D model on the right side, nearly the whole surface seems to be broken. Since these micro-defects are no longer recognizable after the voxelization process, the classification would not be comprehensible for our voxel-based solution. Therefore, we manually change the corresponding label from non-manufacturable to manufacturable in the cases where only small errors exist (like shown in Figure 8.3 left) or remove the 3D model completely from our test data when the 3D model has major errors (like shown in Figure 8.3 right).

From the 4931 3D models, 3266 have been labeled as non-manufacturable and 1665 as manufacturable. From the 3D models labeled as non-manufacturable, 586 have been relabeled as manufacturable and 141 have been completely removed due to major errors. Based on the combination of automated and manual labeling, we have created a data set consisting of 2539 non-manufacturable and 2251 manufacturable 3D models. This labeled

data set is used for the generation of training data files which are processible by the HCNN model.

8.1.4 Experimental Setup

Based on the labeled data set which we have generated as explained in the previous sections, we carried out our experiments. In this section, we will describe the generation of the final experimental data which is generated based on the labeled data and the experimental parameters.

HST Generation

Using the labeled data sets described in Section 8.1.3, we can generate the input data which is processible by the HCNN model. As explained in Section 6.4, there are two different variants for storing the geometric information of the original 3D models as voxel representation: In the first case, the voxels which are intersected by a triangle corresponding to the surface of the 3D model are filled with ones, all others remain zero. For the second case, the X-, Y- and Z-component of the normal vector corresponding to the triangle intersecting a voxel are stored in three separated channels of the voxel representation. With that procedure, more information is stored in the voxel model. Based on first tests, we have decided to use this more complex representation, since the performance of the HCNN is slightly better. These voxel models are used to generate the hst-files. Using the method explained in Section 6.4, we generate 14 different hst-files for each 3D model. As a result, we created $2539 * 14 = 35.546$ instances of non-manufacturable and $2251 * 14 = 31.514$ manufacturable hst-files.

This data set is split into training, validation and test data which are disjunct from each other. For both, the test and the validation set, we are using 200 manufacturable and 200 non-manufacturable 3D models respectively the corresponding hst-files. Therefore, 2139 non-manufacturable and 1851 manufacturable 3D models can be used for the training. Using that data sets, we carried out our experiments.

Experiment Parameters

Since our solution is based on the two main blocks HCNN for the classification of the 3D models into manufacturable or non-manufacturable and the LRP extension for the visualization of critical geometries, we will define the parameters of the two building blocks in this section.

HCNN

For our experiments, we used the HCNN architecture as described in Section 6.2. For the network setup, we used the standard architecture used by the authors for classification tasks. The architecture consists of a data layer for the data access followed by seven convolutional blocks which consist of a convolution layer with using a Xavier weight initializer [35], a batch normalization layer, a scale layer, a ReLU activation layer and a max pooling layer. Using that convolutional blocks, the data representation is down sampled from 512^3 to 8^3. The convolutional blocks are followed by a fully connected block consisting of a dropout layer with a dropout ratio of 0.5, a fully connected layer using the Xavier weight initializer and 128 outputs, a batch normalization layer, a scale layer and a ReLU activation layer. We treat the problem of manufacturability analysis as a classification problem with the two classes manufacturable and non-manufacturable. Therefore, the last block again consists of a dropout layer with dropout ratio 0.5 and fully connected layer using the Xavier weight initializer and two outputs for the two classes. For the training phase, a softmax loss layer is used, for the test phase an accuracy layer.

We trained the HCNN instances on NVidia GTX1080Ti GPUs with 11 GB memory provided by the Paderborn Center for Parallel Computing. Even though the authors of the HCNN approach state in their paper that training with a batch size of 32 is feasible when using 512^3 voxels, we could only train the HCNN model using a batch size of 16. When training with larger batch sizes, the memory capacities were exceeded. We trace this back to the sometimes very complex 3D models we are using and the associated higher memory requirements for those 3D models. We trained the HCNN model using different parameterizations for the following hyper parameters: base learning rate, learning rate policy (exponential or step-wise decay) and the corresponding decay value and weight decay for regularization. We tuned that hyper parameters for achieving the highest accuracy on correctly classified instances using the validation set.

LRP

As explained in Section 6.3.1, the LRP approach offers several different rules which can be used for the generation of the visual feedback. Since different choices for applying the LRP rules for the different layers used by the HCNN architecture may have large effects on the quality of the visual feedback, we have chosen to rely on the most recommended rule for each layer [60]. Therefore, we are using the LRP-ε rule for the fully connected layers and the LRP-$\alpha\beta$ rule for the convolutional layers. For the pooling layers, the relevance is passed

to the neuron with the maximum activation in the receptive field of the kernel in the forward step.

8.2 Evaluation

Based on the realization described above, we can begin with the evaluation of our solution. The experiments which we carried out, are designed to verify the capability of our solution to learn geometric features which are decisive for the manufacturability of components. We are evaluating the performance of our approach with regard to the requirements $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ which we defined in Section 3.1.3. For the experiments, we are using the synthetic data set described in Section 8.1, which is labeled on the example of the *minimum wall thickness* criterion. Using that data set for our experiments, our main focus is on the two requirements $R_{2-ma} \mathrel{\widehat{=}} accuracy$ and the quality of the $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ regarding critical geometries. Using these criteria, we will first show that our solution is capable of learning to evaluate 3D models with respect to the *minimum wall thickness* criterion and subsequently generating visual feedback. Subsequently, we are investigating the criteria $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$.

8.2.1 Classification Accuracy

The requirement $R_{2-ma} \mathrel{\widehat{=}} accuracy$ is our primary criterion to begin with. Based on this, it must first be shown that our solution is able to learn to successfully classify 3D models as manufacturable or non-manufacturable based on the created data set. The Tables 8.1 and 8.2 show the generated results. The individual columns show the number of true positives and negatives as well as false positives and negatives and in the last column the classification accuracy. Since our solution system is to be trained for further manufacturability criteria based on production data which is currently collected and labeled at the AM service provider which we use as reference, it is important to investigate how the number of 3D models used for the training affects the performance of our solution. We therefore trained the HCNN instances with 200, 400, 600, 800, 1600 and 2400 different 3D models. In all of those cases, the data sets are balanced, i.e. they consist of 50% manufacturable and 50% non-manufacturable components.

Additionally, we compared the classification accuracy for the two different variants for the calculation of the final decision which we explained in Section 6.5. The first variant

Table 8.1 Experimental results: true and false positives, true and false negatives and the classification accuracy in percent for different settings.

	Single/joint decision	True positive	True negative	False negative	False positive	Classification accuracy
200 Instances	single	65.11%	68.00%	34.89%	32.00%	66.55%
400 Instances	single	80.82%	56.32%	19.18%	43.68%	68.57%
600 Instances	single	69.07%	86.61%	30.93%	13.39%	77.84%
800 Instances	single	74.82%	88.18%	25.18%	11.82%	81.50%
1600 Instances	single	71.64%	94.07%	28.36%	5.93%	82.86%
2400 Instances	single	73.00%	94.75%	27.00%	5.25%	83.88%

considers each hst-file as a single instance, the second variant uses all 14 hst-files belonging to a 3D model to calculate a joint decision.

Table 8.1 shows the results of our experiments for the variant of using an individual decision for each hst-file. The amount of training instances for each experiment is shown in the left column and the following column shows the type of decision making: Single decision for each test instance or joint decision for all 14 orientations corresponding to a 3D model. In our experiments, 3D models which are non-manufacturable are considered as the positive class. *True positive* means that the HCNN model rated a 3D model as non-manufacturable correctly, *true negative* represents 3D models which have correctly being classified as manufacturable. *False negatives* represent non-manufacturable 3D models which are classified as manufacturable and *false positive* means that a 3D model is rated as non-manufacturable while being manufacturable.

First of all, we will deal with the effect of different amounts of training data. The results presented in Table 8.1 show a clear dependency of the classification accuracy on the amount of training instances which have been used. The best result of 83.88% has been achieved using 2400 training instances. The HCNN instances trained on 800 and 1600 training instances can still achieve comparable results with an accuracy of 81.50% respectively 82.86%. For less than 800 training instances, the classification accuracy starts decreasing significantly. Using 600 instances, an accuracy of 77.84% can be achieved, for 400 and 200 instances, the classification accuracy decreases to 68.57% respectively 66.55%. These results show, that already for the recognition of the relatively simple *minimum wall thickness* feature, several hundred training instances have to be used, to enable the HCNN model to learn to recognize the decisive feature with a relatively high accuracy.

A closer look at the results shown in Table 8.1 reveals an interesting effect. In the cases where 800 or more training instances have been used, the false positive rate is significantly

Table 8.2 Experimental results: true and false positives, true and false negatives and the classification accuracy in percent for different settings and joint decision.

	Single/joint decision	True positive	True negative	False negative	False positive	Classification accuracy
200 Instances	joint	95.00%	20.50%	5.00%	79.50%	57.75%
400 Instances	joint	96.50%	16.00%	3.50%	84.00%	57.25%
600 Instances	joint	94.00%	52.50%	6.00%	47.50%	73.25%
800 Instances	joint	92.00%	64.50%	8.00%	35.50%	78.25%
1600 Instances	joint	91.50%	72.00%	8.50%	28.00%	81.75%
2400 Instances	joint	90.00%	73.50%	10.00%	26.50%	84.25%

lower than the false negative rate. In most cases, the *false positive* rate is lower than 10% while the *false negative* rate is between 25 and 30%. This effect is caused by the fact that critical geometries with too thin wall thicknesses are sometimes "overlooked" due to the rotational variance of the HCNN model. The HCNN model is not able to detect the critical geometries in some orientations. This results in the relatively high *false negative* rate. In contrast, the *false positive* rate is very low. This means that 3D models are only very rarely incorrectly evaluated as non-manufacturable while being manufacturable. Since the high *false negative* rate is mainly caused by the rotation variance, we have designed the second variant of the joint evaluation to reduce the high number of *false negatives*.

As can be seen in Table 8.2, there are clear differences between the results of using a single decision and the joint decision. The maximum accuracy which is achievable is relatively similar. Using the single decisions, a maximum accuracy of 83.88% can be achieved while for the joint decision a maximum of 84.25% is reached. Major differences can be seen when considering the rates of true/false positives/negatives. For the single decision experiments, it can be seen that the *false positive* rate is extremely low while the *false negative* rate is relatively high. As explained above, the low rate of *false positives* arises due to the fact that manufacturable 3D models do not have a critical feature and only very rarely a critical feature is falsely detected in the 3D models geometry. The high false negative rate arises from the fact that the critical features can not be detected in all orientations by the rotation-variant HCNN model. Therefore, achieving a *true positive* rate very close to 100% and accordingly a *false negative* rate close to 0% is simply not possible. On the other hand, when using the joint decision, this effect develops exactly in the opposite direction. The *false positive* rate is growing while the *false negative* rate is significantly decreasing. This effect is caused by the fact that a 3D model is already evaluated as non-manufacturable if a critical feature is detected for only one of the 14 orientations. As a result, falsely detected critical geometries

have a greater impact on the *false positive* rate. On the other hand, however, significantly fewer critical geometries are overlooked by the HCNN model since the detection for one of the 14 orientations is sufficient. The joint decision can achieve a false negative rate in the single-digit range. This setup is useful when a very high *true positive* rate should be achieved and thus no or hardly any critical geometries are missed.

The results show that our HCNN-based solution is able to detect critical geometric features in 3D models with a relatively high accuracy. In the next section, we will analyze the quality of our solution with regard to the requirement $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$. The visual feedback generated by our solution is not only an additional information which customers of AM service providers can use to modify their 3D models if necessary, but also it is an extreme support for understanding the decisions of the HCNN model. Taking a deeper look on correct and wrong decisions of the HCNN model enables us to interpret what the HCNN model is capable of and what it is not.

8.2.2 Visual Feedback

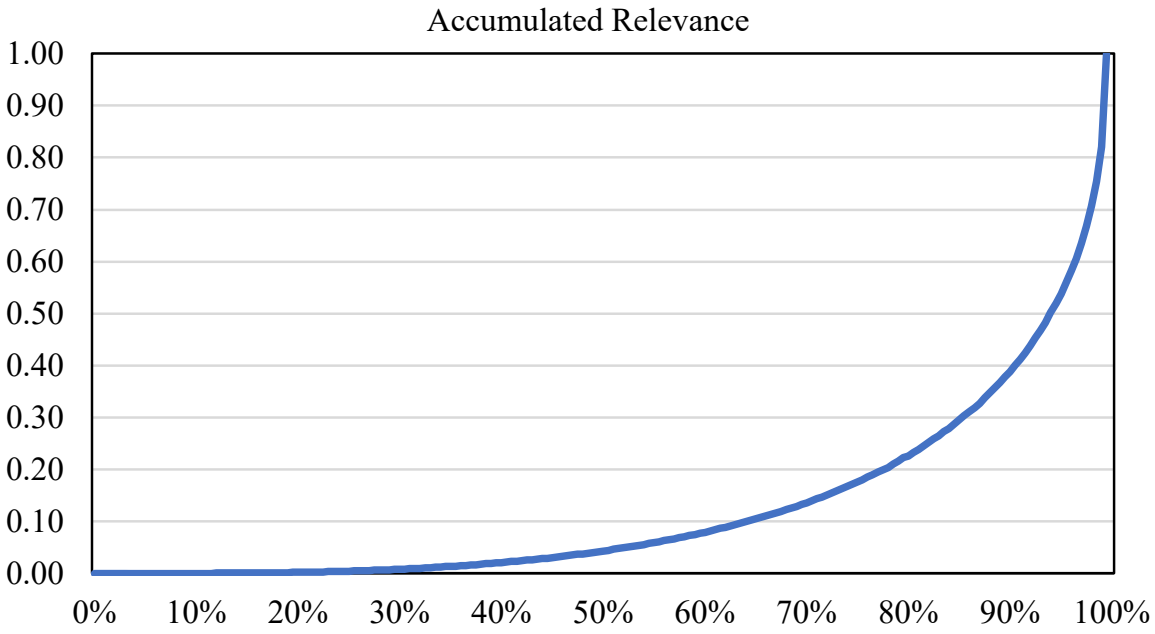


Fig. 8.4 Relevance accumulated over the voxels.

Using the LRP approach, which we described in Section 6.3.1, we are able to visualize the relevance of the input voxels for the decision of the HCNN model. Using that method, we are able to verify, if our solution is generating its decisions based on the correct ground truth features. As already described in Section 8.1.3, the HCNN model only receives the geometric

information and the label manufacturable or non-manufacturable belonging to the 3D model in the training process, but no information regarding the problematic sub-geometries within a 3D model. With the help of the visual feedback, we can show that the HCNN model is nevertheless able to extract the correct geometric features from the data on its own.

In general, the LRP approach is propagating the relevance back from the output of the HCNN to the input. Using that procedure, for each input voxel a relevance score is calculated. That results in a heatmap visualization. The voxelized models often consist of a million or more voxels. For being able to generate a clearer visualization, we therefore filter the voxels with the highest relevance. Since most of the voxels have nearly no relevance for the decision of the HCNN model, the majority of the voxels can be discarded for the creation of the visualization. Figure 8.4 shows the distribution of the relevance over the voxels corresponding to the example visualization shown in Figure 8.5. The voxels are sorted by their relevance and subsequently the relevance is accumulated over the voxels. As can be seen, the 50% of the voxels with the lowest relevance contain only about 4% of the relevance for the decision. In contrast, the 5% of the voxels with the highest relevance contain around 50% of the overall relevance. By removing the voxels with a low relevance, we are able to generate the visualization much faster and at the same time create a clearer visualization.

Based on a series of experiments, we concluded that filtering out all voxels with at least 1% of the relevance of the voxel with the highest relevance leads to an expressive visualization. Including more voxels leads to a higher computation time and does not cause a notable change on the visualization. After filtering out the voxels with the highest relevance, we use a Gaussian smoothing kernel to smooth the visualization. The calculation of the values for each kernel element $\mathbf{x}_i$ are shown in Equation 8.1. We have chosen a kernel size of 5 and $\sigma = 5$. This is, of course, a subjective assessment and serves to clarify our results in this thesis. For a more extensive use of the approach, this representation can be further revised.

$$K(\mathbf{x}_0, \mathbf{x}_i) = \exp\left(-\frac{(\mathbf{x}_0 - \mathbf{x}_i)^2}{2\sigma^2}\right) \tag{8.1}$$

Two examples for non-manufacturable 3D models, which have been classified correctly, are shown in Figure 8.5 and 8.6 together with the ground truth data. In the LRP visualization on the left, the relevance in marked in shades of red. The higher the relevance for the decision is, the darker is the shade of red. As can be seen, the most relevant areas of the 3D model for the decision of the HCNN model correspond to the critical geometries of the ground truth data. The HCNN model has correctly identified most of the non-manufacturable features of the two example 3D models. As can been, for the 3D model shown in Figure 8.5, the larger

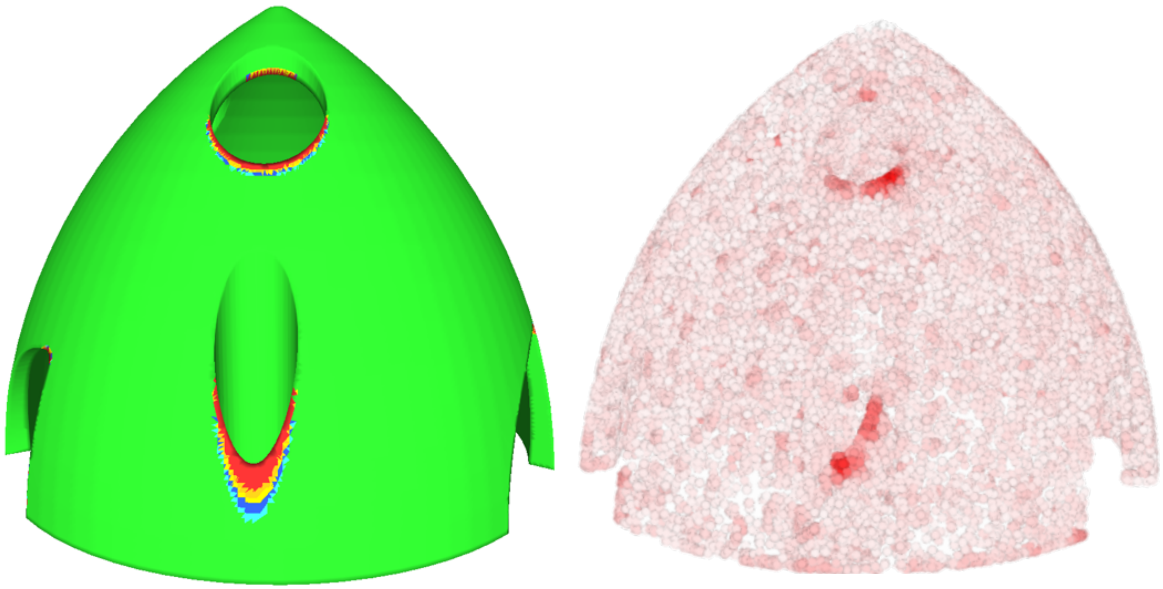


Fig. 8.5 The ground truth data of an example 3D model (left) and the corresponding output generated by the LRP approach (right).

critical areas are detected. However, due to the rotation variance, the focus is on the right side of the critical area. Also in the second example shown in Figure 8.6, the areas around the eyes of the robot head are clearly identifiable as the most relevant. Only the small part at the top of the notches in Figure 8.5 can be identified as critical only to a limited extent. These areas are close the limit of being non-manufacturable. The effect that these type of non-manufacturable geometries close to the limits of the manufacturability definition, can also be seen for other 3D models. It can be attributed to the labeling of the training data respectively the definition of the *minimum wall thickness* criterion itself, since the definition is relatively fuzzy.

An explanation of the fuzzy labeling is given in Figure 8.7, which is showing the ground truth data of a 3D model labeled as non-manufacturable. The HCNN model has incorrectly classified the 3D model as manufacturable. As can be seen, only a small edge on the right side of the 3D model is labeled as non-manufacturable geometry. The edge has an angle only slightly beyond 90 degree. These cases represent the edge cases of the manufacturability definition for the *minimum wall thickness* criterion. Depending on the orientation of the 3D model, they are partially evaluated as manufacturable and partially as non-manufacturable. This effect can be seen within this example 3D model itself. The edge on the right side of the 3D model is rated as non-manufacturable. In contrast, the edge on the left side, which is a mirrored variant of the edge on the right side, is labeled as manufacturable. Since these types of problem geometries are not completely uniformly classified even in the labeled

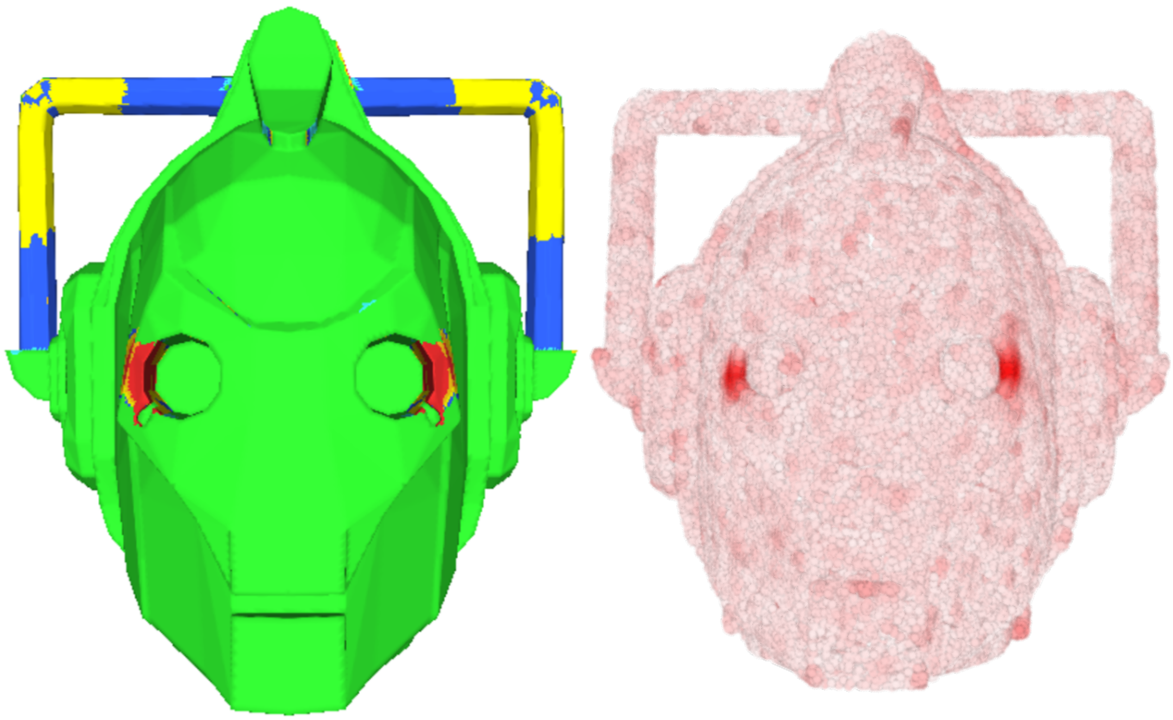


Fig. 8.6 The ground truth data of a second example 3D model (left) and the corresponding output generated by the LRP approach (right).

training data, misclassifications of these geometries by the HCNN model can sometimes not be avoided in these situations.

The LRP system can also be used to visualize the relevance distribution of the HCNN model for a 3D model which has been classified as manufacturable. Figure 8.8 shows the LRP visualization of a 3D model, which has been correctly classified as manufacturable. In contrast to a 3D model that is classified as non-manufacturable, the LRP approach does not show any concrete geometries that are decisive for the decision when a 3D model has been evaluated as *manufacturable*. As already explained in Section 8.2.1, in such cases there is no concrete partial geometry that is relevant for the decision. The HCNN approach thus considers all areas of the 3D model to be relevant, relatively evenly distributed.

The presented results show that the HCNN model is able to learn the manufacturability criterion *minimum wall thickness* with a high accuracy of about 84% based on the labeled data sets. Using the LRP extension, we are additionally able to verify that the HCNN model makes its decisions based on the correct geometric features and is able to generate a visual feedback with a relatively high resolution. Therefore, based on the current state of our solution, we rate the performance of our solution with respect to the two requirements $R_{2-ma} \mathrel{\widehat{=}} accuracy$ and $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ as good. Nevertheless, there are still some optimization potentials,

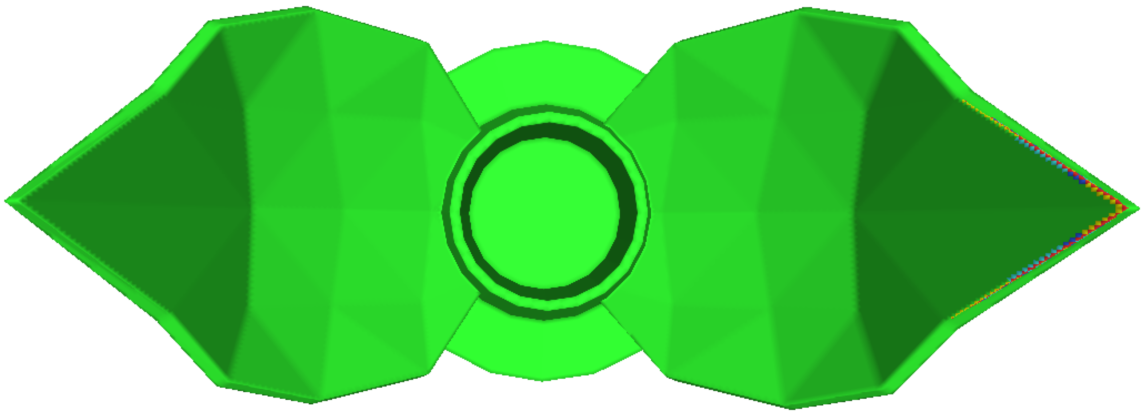


Fig. 8.7 Ground truth data of a 3D model which is non-manufacturable, but has been misclassified as manufacturable by the HCNN model.

which we want to address in Section 8.3. Previously, we will investigate the performance of our solution with regard to the remaining requirements in the next section.

8.2.3 Secondary Requirements

Subsequent to our consideration of the requirements $R_{2-ma} \mathrel{\widehat{=}} accuracy$ and the quality of the $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$, we are going to examine the remaining requirements $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ in this section.

The requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$ can only be evaluated to a limited extent at the current time, as our solution has not yet been trained to evaluate other manufacturability criteria. Since we have basically shown that the approach is able to learn a geometric criterion with the help of appropriate training data, it should in principle be possible to learn further manufacturability criteria with suitable data, which would provide a high *adaptability*. Currently, production data is collected and labeled at the AM service provider which we use as reference. Since our experiments have shown that several hundred 3D models that are positively labeled with respect to a manufacturability criterion under consideration are necessary to learn a geometric criterion, there is not yet sufficient data for further experiments at the current state. As soon as a promising amount of data is available, we will conduct further experiments based on the collected data to prove the *adaptability* of our solution.

A high *data representation detail* is generally provided by our solution, which can use up to 512^3 voxels. This allows even filigree details of the 3D models to be represented. Due to the high detailed resolution, however, the maximum size of the 3D models is limited to a diameter of 102.4 *mm*. As described in Section 8.1.3, we have chosen a wall thickness

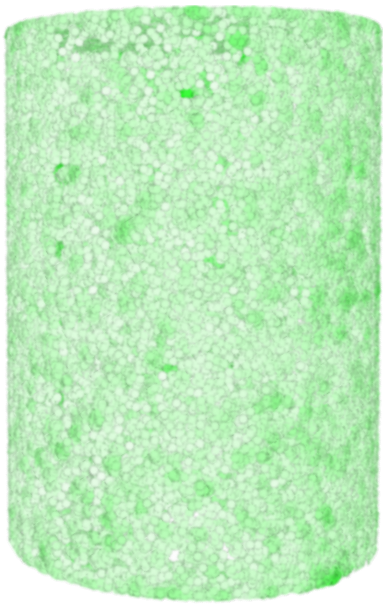


Fig. 8.8 LRP visualization of an example 3D model which has been correctly classified as manufacturable.

threshold smaller than the physical thresholds of the AM machines used at the AM service provider, for being able to generate a balanced training data set. For SLM, the real threshold value is around 0.6 *mm* depending on the machine, for SLS even about 1.0 *mm* [2]. Therefore, the resolution can be increased especially when processing SLS components and thus the maximum component size that can be processed, can be increased. Additionally, for further manufacturability criteria which are described by larger contexts within the 3D models, the resolution could also be increased, since extremely small details may not be relevant for those kind of manufacturability criteria. By increasing the edge length of the used voxels from 0.2 to 0.4 *mm*, 3D models with a diameter up to 204.8 *mm* could be processed. For the analysis of manufacturability criteria which are rather global in nature, such as the manufacturability of cooling channels, a resolution of 0.4 *mm* is sufficient. The data representation can therefore be adjusted for each criterion. As a result, we rate the quality of our solution with regard to the criterion $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$ as good.

The last requirement we are considering is the requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$. The *Manufacturability Analysis* in the process chain of AM service providers should take a maximum of a few minutes, ideally only a few seconds, to enable a smooth ordering process for the customer. The process of generating an evaluation of a 3D model regarding its manufacturability including the visualization of critical geometries using our solution consists of several steps.

The process starts with the generation of the hst-files which the HCNN model is using as input data. As explained in Section 6.4, the generation of the hst-files again consists of several steps. Based on the STL file which has automatically been generated from the 3D model which the customer has uploaded, the point cloud representation is generated using the virtual scanner. With the current implementation, the creation of the point cloud with ray arrays consisting of 1024^2 rays, takes on average two minutes per 3D model. By raytracing the six perspectives in parallel, this time could be reduced to 20 seconds. Subsequently, the hst-files itself are generated. On average, this takes 15 seconds per hst-file. Therefore, using parallelization, the input data for the HCNN can be generated in about 35 seconds per 3D model. For the generation of the hst-files, a standard laptop hardware with an Intel core i7 processor, 16 GB of RAM and an NVidia GTX1050 GPU has been used. Therefore, using more powerful hardware, the time required to generate the data can be further reduced.

The actual classification of a 3D model using the HCNN model takes 18 seconds on average for a batch including the 14 different orientations generated for each 3D model using an NVidia GTX1080Ti GPU. Therefore, the calculation of the prediction if a 3D model is manufacturable or not, can be generated in less than a minute. Subsequently, the visualization is generated. The raw relevances calculated by the LRP method are generated in 12 seconds on average. Based on these, the final visualization is generated. In the current state, only an experimental visualization is generated. Creating the visualization takes between one and two minutes dependent on the amount of voxels of the 3D model. Compared to the previous steps, the computation time for the creation of the visualization is relatively long. This is mainly caused by the smoothing of the relevances for the optimization of the visualization. As an alternative, we have generated another visualization that can be generated much faster.

An example of the alternative visualization for the same 3D model, which has already been shown in Figure 8.5, is shown in Figure 8.9. Voxels which are rated as highly relevant for the decision of the HCNN model are marked in blue, all others in green. It is generated by filtering out the 0.2% of the voxels with the highest relevance. Subsequently, we perform a distance-based outlier removal to further simplify the representation. This clearly highlights the main features. A drawback of this alternative is that this procedure removes some of the voxels that are relevant for the decision of the HCNN model. It only keeps relevance clusters. The benefit of this alternative is that this visualization can be generated within 15 to 20 seconds, depending on the 3D model.

Therefore, the entire process from uploading a 3D model to the finished visualization takes about 100 seconds for the second alternative. However, since the current variant of our solution is developed for experimental purposes, it has not been optimized in terms of computation time. If integrated into the process chain of AM service providers, an

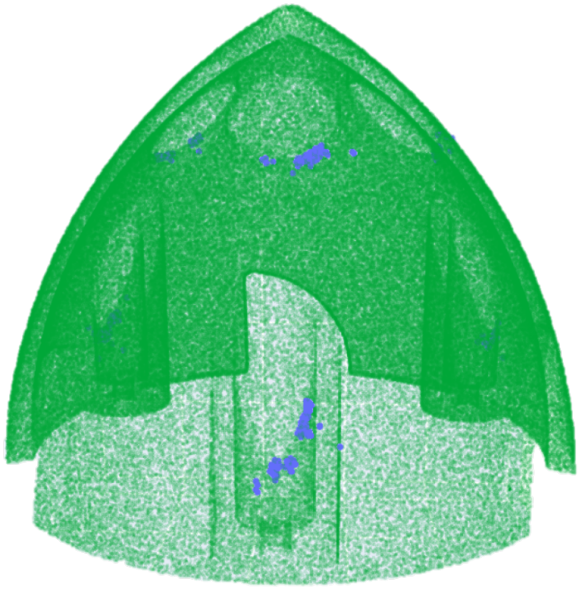


Fig. 8.9 Alternative visualization for the example 3D model already shown in Figure 8.5.

optimization can be performed, which should enable a further reduction of the process time. Even now, however, it can already be stated that the time required to generate a prediction regarding the manufacturability of a 3D model, including the generation of a visual feedback, is within the defined requirement of a few minutes. In its current state, we therefore rate the performance of our solution with regard to the requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ as fair. Nevertheless, we are confident, that we can achieve a faster computation time through an optimization of the whole process with regard to the computation time.

Since the HCNN system only needs to be trained once for each manufacturability criterion, the training time has no effect on the customers' ordering process. However, it may be useful to train the system again if, for example, a new AM machine that has different specifications is integrated into the machine park. The training of the HCNN instances on the different data sets of our experiments took between 24 and 36 hours to achieve the best validation accuracies. Therefore, it is possible to train the HCNN system e.g. for further machines within one and a half days.

Table 8.3 is showing a comparison of our solution with the solutions presented in Section 5.1. As can be seen, our solution is the only solution which provides the necessary adaptability while achieving a good accuracy and visual feedback based on a suitable data representation. Only the computation time is at the current state only slightly within the limits of the defined requirements. Thus, based on the ideas of the existing solutions and the state-of-the-art

Table 8.3 Comparison of our solution with the currently used industrial tools, the design rule-based research approaches and simulation-based approaches for *Manufacturability Analysis* in the AM domain, the ML-based approach of Balu et al. from a domain of bore drilling.

Method/Metric	R_{1-ma} Adaptability	R_{2-ma} Accuracy	R_{3-ma} Suitable Data Representation	R_{4-ma} Visual Feedback	R_{5-ma} Computation Time
Industrial Tools	– –	0	0	+	0
Rudolph and Emmelmann	– –	No statement possible	+	+	– –
Shi et al.	0	No statement possible	+	++	No statement possible
Tedia and Williams	– –	No statement possible	+	+	–
Simulation-based	+	++	++	+	– –
Balu et al.	++	+	–	–	+
Our Solution	++	+	+	+	0

methods of 3D DL, we have designed a solution that combines most of the positive aspects of the existing solutions.

In the course of the experiments, however, potentials for improvement have also become apparent. We will discuss these in the following section.

8.3 Discussion

Our experiments have shown that our HCNN and LRP-based solution is able to classify unknown 3D models into the categories manufacturable and non-manufacturable with regard to the *minimum wall thickness* criterion with an accuracy of about 84%. The main factors which limit the accuracy are inaccurate labeled data and the rotation variance of our solution.

The inaccurate data labeling arises from the combination of automated labeling using a software tool and manual relabeling. Since the labeling tool itself is rotation variant, inconsistent labeling occurs especially for edge-case 3D models that are close to the limits of manufacturability. Although many errors were corrected by manual relabeling, it is not possible to manually examine each 3D model in such detail that an incorrect label can be completely prevented for the large amount of 3D models. Since the database is not 100% correctly labeled, the maximum classification accuracy of our solution is limited, too. The

data currently collected at the AM service provider will provide a better quality, as it will be generated in the context of the real production. Data of non-manufacturable components is generated either when a defect actually occurs during the production of a component or when the AM engineers assess that a component is non-manufacturable during the preparation of the corresponding build job. Each 3D model is therefore verified in detail by an AM expert and thereby labeled with high confidence.

The rotational variance arises from the rotation-variant convolution operation that the HCNN uses. By reusing the prediction of the HCNN model calculated for each individual orientation of a 3D model for the calculation of a joint decision for the 14 orientations corresponding to a 3D model, we can at least partially circumvent this property. However, the joint evaluation leads to a significantly higher false positive rate. An alternative way to significantly reduce the rotation variance, is to adapt the HCNN approach to a multi-view system similar to the RotationNet approach, which we presented in Section 7.2. Instead of evaluating the decisions of the HCNN model jointly for all orientations of a 3D model subsequent to the actual prediction generation of the HCNN model, the HCNN model can also be adapted to evaluate all 14 orientations of a 3D model jointly. If all 14 orientations belonging to a 3D model are provided to the HCNN model as a collective instance, significantly more information is available for the classification. The collective use of all 14 views would lead to a significant reduction of the rotation variance, which could potentially further increase the classification accuracy. Therefore, we are confident that we could even surpass our current results by changing the HCNN architecture in the described manner.

8.4 Summary

Our experiments have confirmed that a 3D-CNN-based approach is able to learn a geometric criterion that is crucial for the manufacturability of 3D models, based on a correspondingly labeled data set. Besides just deciding whether a 3D model is manufacturable or not, our solution is also able to generate visual feedback, based on which we could validate that indeed the correct discriminative criterion is learned. In addition to the validation that the HCNN model has learned the correct criterion, the visual feedback can be used by customers of AM service provider to modify non-manufacturable 3D models based on the generated visualization. Therefore, we can state that our solution provides a good quality with regard to the requirements $R_{2-ma} \mathrel{\widehat{=}} accuracy$ and $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$. Since our goal is to develop a solution that can be integrated into the processes of AM service providers, the requirements defined in Section 3.1.3 had to be met.

Based on the experiments, it could be shown that the HCNN-based approach is able to learn a relatively simple geometric criterion based on a few hundred non-manufacturable 3D models. Form this fact, it can be concluded that our solution is assumed to be able to learn further manufacturability criteria based on a manageable amount of production data. Another important factor is that our solution can be trained based on the raw geometric data only using the corresponding label, but without additional information regarding concrete critical sub-geometries within the 3D models. Thus, no further effort is required to mark concrete partial geometries in the 3D models when creating further data sets.

Additionally, the requirements $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ and $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$ are met by our solution. The current experimental implementation is able to generate the evaluation including visualization in about 100 seconds on average. Using the resolution of 512^3 voxels, our solution is able to process the 3D models with a high level of detail and, depending on the choice of the voxel size, it can be adapted for different manufacturability criteria to be evaluated. Although the capability of our solution regarding the requirement requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$ can not yet be proven on the basis of the experiments conducted so far, the adaptability is supported by the general data-driven structure of our solution. Prior to providing a final conclusion in the Chapter 10, we will discuss the experiments regarding the process step of *3D Component Recognition* of additively manufactured components in Chapter 9.

Chapter 9

Realization and Evaluation: 3D Component Recognition

In this chapter, we are presenting the realization and evaluation of our developed solution for the *3D Component Recognition* of additively manufactured parts based on appropriately designed experiments. As explained in Chapter 7, our solution is designed for the integration into the process chain of AM service providers. It should enable AM service providers to realize an automated *3D Component Recognition* solution which is able to recognize components produced via the SLS technique build job per build job. In principle, the solution can also easily be adapted for other scenarios, such as collaborative recognition of the entire production of a day. However, the described case initially represents the standard scenario.

In this chapter, we describe the concrete realization of our solution using appropriately design build jobs for demonstration. We will describe the realization in Section 9.1. At first the generation of the build job data sets, which are later on used for our experiments, is described in Section 9.1.1. Since, to the best of our knowledge, no benchmark data sets exist for the recognition of additively manufactured components, we created our own data sets and aligned them to standard build jobs which occur in the daily production of AM service providers. We will describe that step in detail in the following Section 9.1.1. Subsequently, we will describe the generation of the training data based on the defined build job data sets in Section 9.1.2. Additionally, the experimental setup used for our evaluation is described in Section 9.1.3. The evaluation will subsequently be presented in Section 9.2. We will finish this chapter with a discussion and summary in the Sections 9.3 and 9.4.

9.1 Realization

For being able to evaluate the performance of our solution which we explained in Section 7 with regard to the requirements $R_{1-cr} \mathrel{\widehat{=}} \textit{recognition rate}$, $R_{2-cr} \mathrel{\widehat{=}} \textit{robustness}$, $R_{3-cr} \mathrel{\widehat{=}} \textit{adaptability}$, $R_{4-cr} \mathrel{\widehat{=}} \textit{process time}$, $R_{5-cr} \mathrel{\widehat{=}} \textit{scalability}$ and $R_{6-cr} \mathrel{\widehat{=}} \textit{process costs}$, which we defined in Section 3.2.2, we first need to realize a concrete showcase. This means that we perform the steps described in Section 7.1 to implement our solution for concrete data sets. This process starts with the definition of corresponding build job data sets. Based on these data sets, the actual training images have to be generated using the method for the calculation of physically sound orientations, described in Section 7.4.2 and the rendering and augmentation methods described in Section 7.4.3 and 7.4.4. Subsequently, the training of the RotationNet model can start. When the training process is terminated, the inference can start.

9.1.1 Base Data Sets

In Section 7.4, we already explained that our system is designed to recognize the components build job per build job. Therefore, we need the data of real build jobs from AM service providers or very similar data to evaluate the performance of our solution for this purpose. For data protection reasons, we can not use real build job data from the AM service provider which we used as reference. Since there are no benchmark data sets for the recognition of additively manufactured components, yet, we decided to create our own data sets for the evaluation.

In order, to design the data sets as realistic as possible, we have analyzed the build jobs which have been produced in the last years at the AM service provider we used as reference. The SLS machines, which are used by the AM service provider, are among the most commonly used machines in the industry. Also, the machines of other manufacturers mostly have similar specifications and therefore similar build job characteristics. Therefore, the data sets, which we have generated, are generally representative for all common SLS machines.

However, there is not only one standard configuration for the build jobs for SLS. Especially, with regard to the amount of different components produced in a single build job, the build jobs often differ very much. Most of the build jobs produced by AM service providers contain between ten and 100 different components. Since the individual components belong to many different customers and therefore mostly come from different domains, there are usually not too many geometrically similar components in one build job. Therefore, we generated three different build jobs which are composed of 30, 50 and 100 different randomly selected 3D models corresponding to the components which have to be recognized after the

training phase of the system. For the explanation of the experimental results in section 9.2, these data sets are named Random30, Random50 and Random100.

An important aspect for the generation of a suitable reference data set is that the components must match the domain of AM. They should therefore represent geometries that resemble the components produced in the real production of AM service providers. For this purpose, we have used the free database Thingi10K [152]. The Thingi10K data base contains 10000 3D models corresponding to the AM domain. Therefore, it is perfectly suited for our needs.

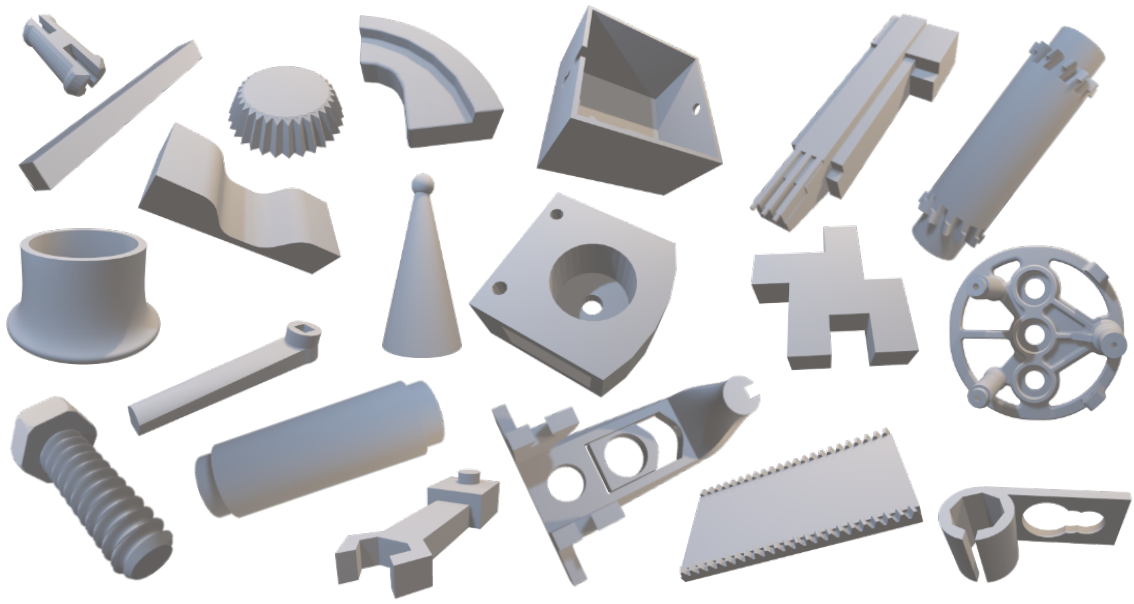


Fig. 9.1 Example 3D models chosen from Thingi10K [152] for the randomly generated evaluation data sets.

Figure 9.1 shows a subset of example 3D models which has been randomly chosen from the Thingi10K data base. As can be seen, most of them are relatively different to each other and at least for humans easily distinguishable. With our evaluation, we want to assess the performance of our solution for both simple and difficult data sets. The differentiation of objects becomes more difficult if either more different objects are considered or the objects are much more similar. The three data sets which we generated so far, only show different difficulties due to the varying number of 3D models. In order to include the aspect of similarity, we have generated three additional data sets with ten, 30 and 50 different 3D models each, which are partly very similar. Although, it is unlikely that these data sets will occur in the real production at AM service providers, they are very well suited to evaluate

the performance of our approach. For the explanation of the experimental results in Section 9.2, these data sets are named Similar10, Similar30 and Similar50.

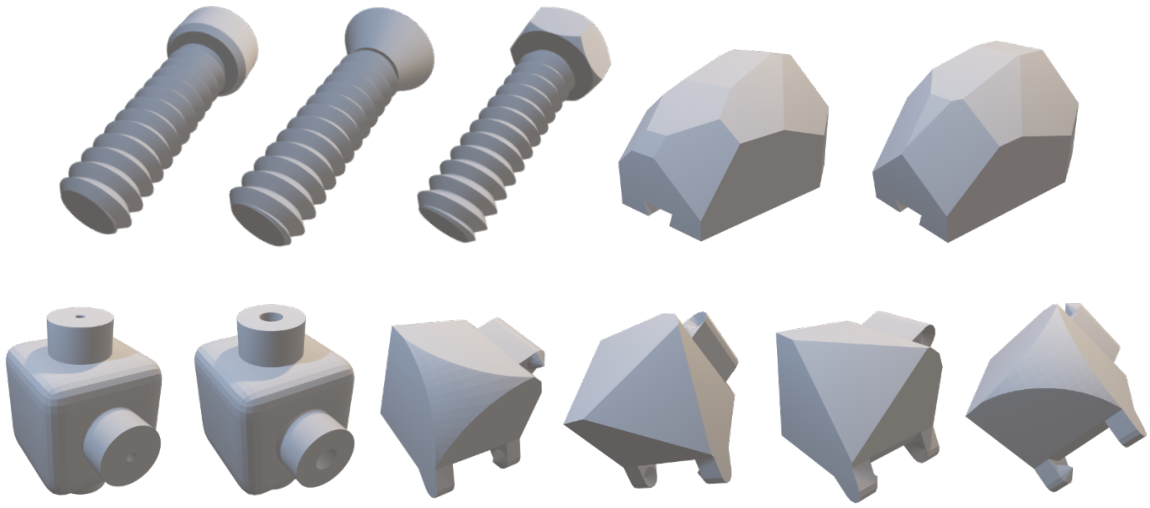


Fig. 9.2 Example 3D models chosen from Thingi10K [152] for the creation of data sets with high geometric similarities between the 3D models.

As can be seen, the 3D models shown in Figure 9.2 are much more difficult to distinguish. Especially, the four parts in the bottom right are even for humans not immediately distinguishable. We conducted our evaluation based on these six data sets. In order to make the data sets usable for the MVCNN based approach RotationNet, the actual training data must first be generated on the basis of the raw 3D models. This process is described in the following section.

9.1.2 Training Data Generation

As already explained in Section 9.1.1, no benchmark data sets exist for the task of recognizing additively manufactured components. Since we want to compare our approach of using physically sound training data with a baseline, we are additionally creating a second training data set using randomly generated orientations for each of the six base data sets described in Section 9.1.1. This idea is based on the fact that only randomized orientations can be used without further assumptions regarding physically sound orientation. We consider our approach of physically sound training data as an optimized geometrical augmentation technique to improve the quality of training data which uses the knowledge of the physical situation for optimization. To be able to validate the effect of our physically sound training data generation, we compare the performance of our solution with the randomly generated

training data augmented using the photometric augmentation techniques described in Section 7.4.4.

Physically Sound vs. Random Orientations

For achieving a fair comparison between the randomly generated and physically sound training data, we are using the exactly same parameters for the generation of the training data for both variants. For each 3D model of the six base data sets, we generate two variants of training data sets. The first contains 100 instances per 3D model, the second one 300 instances.

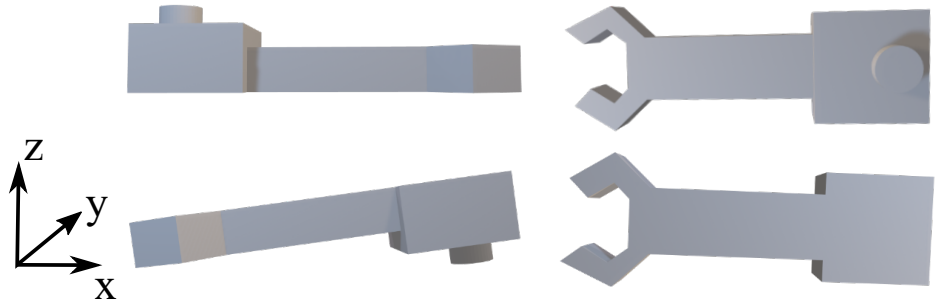


Fig. 9.3 The four calculated physically sound base orientations for an example 3D model from a frontal viewpoint.

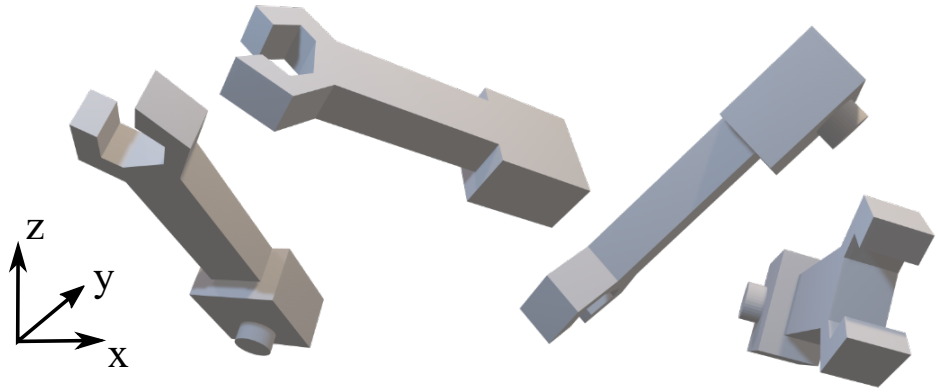


Fig. 9.4 The four calculated physically sound base orientations for an example 3D model from a frontal viewpoint.

This allows us to assess the influence of the amount of training data on the recognition rate. For the physically sound training data, we use the procedure described in Section 7.4.3 for the generation of 100 respectively 300 different training instances. For each 3D model,

we are first calculating the physically sound base orientations and subsequently generate the necessary amount of additional orientations by a rotation around the Z-axis.

For the randomly generated orientations, we just calculate 100 respectively 300 different orientations by calculating randomly distributed points on a sphere and rotate the 3D models in a way the prior Z-axis of the 3D model is pointing in the direction of the calculated point on the sphere. For comparison, Figure 9.3 is showing the four calculated physically sound orientations for an example 3D model and Figure 9.4 is showing four of the randomly generated orientations of the same 3D model.

Image Rendering

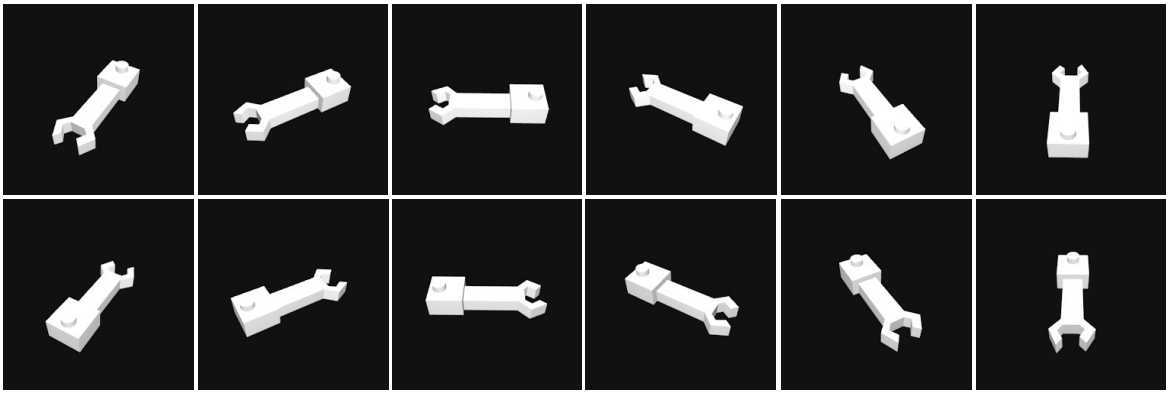


Fig. 9.5 Example of a training instance generated using the physically sound orientations and the camera array configuration v_1.

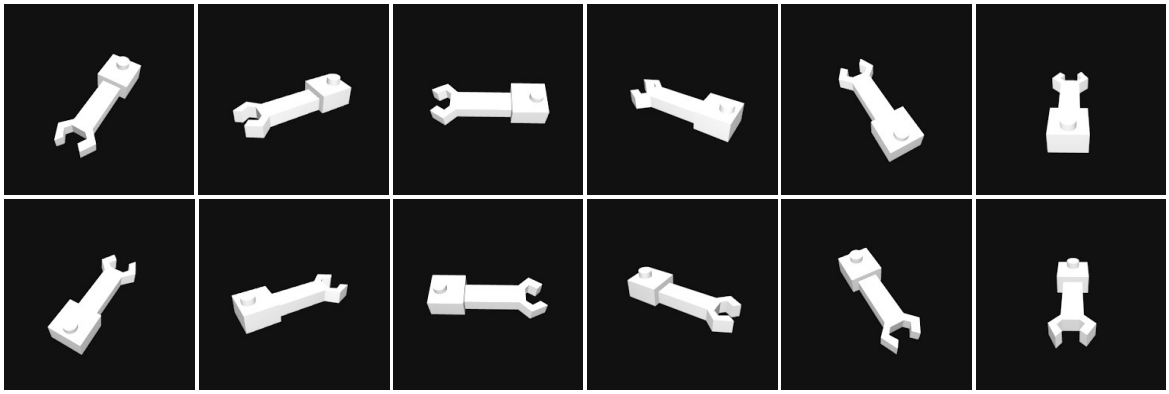


Fig. 9.6 Example of a training instance generated using the physically sound orientations and the camera array configuration v_2.

Based on these orientations, we are rendering 12 images per orientation, using the virtual environment explained in Section 7.4.1. For both, the random and the physically sound case,

the two different camera array variants v_1 and v_2 as described in Section 7.4.1 are used for the rendering. The first is using the fixed vertical angle of 45 degree with respect to the horizontal plane, the second one is using alternating angles of 35 and 55 degree with respect to the horizontal plane.

The second variant is designed to simulate possible inaccuracies in the generation of the physical images. Figure 9.5 and 9.6 are showing an example for the two variants for the physically sound case. Figure 9.7 and 9.8 the corresponding variants for the random case.

Fig. 9.7 Example of a training instance generated using random orientations and the camera array configuration v_1.

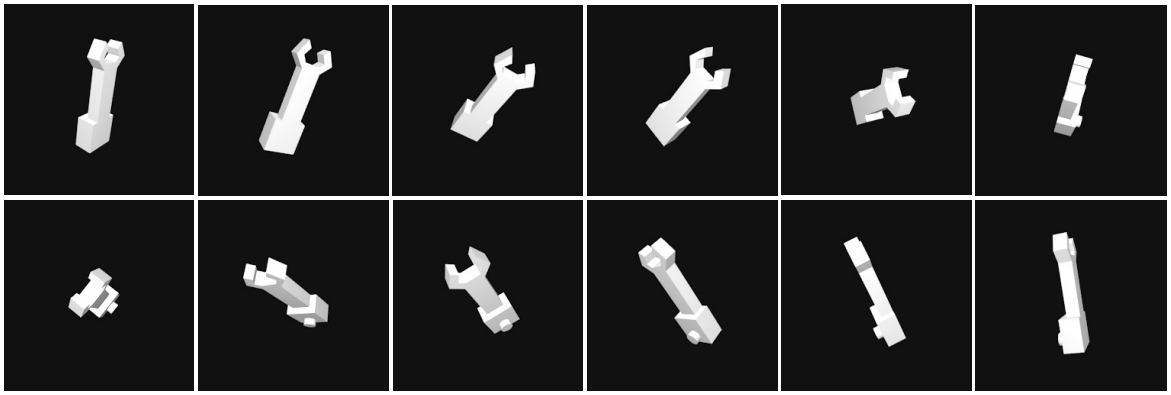


Fig. 9.8 Example of a training instance generated using random orientations and the camera array configuration v_1.

Subsequent to the rendering, the images are augmented using the photometric transformations blurring, RBC, CLAHE, Gaussian noise injection and composition thereof as described in Section 7.4.4. In our experiments, we are comparing RotationNet instances trained using physically sound and randomly generated orientations both trained using the described augmentation techniques and trained without any photometric augmentation.

Training

Based on the different data sets, which have been generated using the steps described above, the RotationNet model can be trained. In case of 100 instances per object, the data is randomly split into 60 training, 20 validation and 20 test instances. In case of 300 instances per object, the data is split into 240 training, 30 validation and 30 test instances. The validation sets are used for early stopping and the snapshot of RotationNet which achieves the best validation accuracy is used for the evaluation on the physical data. We used the standard RotationNet architecture described by Kanezaki et al. [56] and only adapted the network architecture with regard to the amount of different objects for the different data sets. The hyper parameters learning rate, learning rate policy, gamma, step size momentum and weight decay are adapted for each of the different data sets. Based on several experiments for each data set, we have chosen the combination of hyper parameters using which the best results have been achieved for the corresponding data set.

Physical Image Generation

The evaluation of our solution has to be carried out using photos of the physically produced components to be able to assess the capability of our system for the usage in the process chain of AM service providers. Since the recognition station, which we described in Section 7.3, has so far only been designed as a concept and has not yet been physically constructed, an alternative must be used to generate the inference images. We therefore manually photographed the produced components with a smart phone camera. The components have been placed on a black fabric to achieve an optimal contrast to the white components. Each component has been placed in up to six different stable orientations corresponding to its geometry and for each orientation 12 photos have been generated in 30 degree steps around the Z-axis and an elevation angle of approximately 45 degree. For cost reasons, only the two data sets Random30 and Similar50 were physically produced and used for the evaluation. Using the manual image generation, we generated 67 test instances for the Random30 data set and 133 for the Similar50 data set. Therefore, overall $67 * 12 = 804$ images for the Random30 data set and $133 * 12 = 1596$ images for the Similar50 data set have been generated. As explained in Section 7.6, we are using the augmentation techniques RBC, sharpening and a combination of sharpening and CLAHE are used to enhance the quality of the physical images. The influence of this augmentation is evaluated within the experiments.

9.1.3 Experimental Setup

For our experiments, we used the standard configuration of RotationNet for processing images from 12 different viewpoints. The data can be handled to the network by an image data layer which processes 20 batches of 12 images with a resolution of 250 times 250 pixels in parallel. The image data layer is followed by five convolutional blocks. The first two consist of a convolutional layer, a ReLU activation layer, a pooling layer and an Local Response Normalization (LRN) layer, the following three only of a convolutional layer and ReLU activation layer. The convolutional blocks are followed by two fully connected blocks consisting of a fully connected layer, a ReLU activation layer and a dropout layer. The final recognition stage consists of a last fully connected layer and a subsequent softmax loss layer for the training respectively accuracy layer for the validation and test phase. We are using a RotationNet instance pre-trained on on the ILSVRC data set [111] and apply transfer learning via fine-tuning on the network based on our data sets. Therefore, all parameters of the network architecture are the standard parameters provided by the authors of RotationNet [56].

The different RotationNet instances for the different data sets are trained on NVidia GTX1080Ti GPUs with 11 GB Memory provided by the Paderborn Center for Parallel Computing. We used a batch size of 20 what leads to a parallel processing of 240 images per training iteration. During our experiments, we optimized the following parameters: base learning rate, learning rate policy (step wise or exponential decay) and the corresponding value for the learning rate decay and weight decay for regularization. We used the validation set to carry out a validation every 200 iterations for the experiments on the Random30 data set and 1000 for the experiments on the Similar50 data set. For each experiment, the RotationNet state which achieves the best accuracy on the validation set is used for the evaluation on the corresponding test set.

9.2 Evaluation

The main goal of our evaluation is to verify that our approach based on RotationNet and physically sound training data is able to recognize additively manufactured components with a high recognition rate. Additionally, we want to show that our data-driven augmentation technique of using physically sound training data is outperforming standard augmentation techniques. Therefore, we trained the RotationNet model both on physically sound training data sets and randomly generated training data, as explained in the previous section and evaluated the performance on physical images for the Random30 and Similar50 data sets.

We are first dealing with the evaluation of the experiments regarding the requirement $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, since a high recognition rate is the main requirement which has to be fulfilled. Subsequently, we are dealing with the additional requirements $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{3-cr} \mathrel{\widehat{=}} adaptability$, $R_{4-cr} \mathrel{\widehat{=}} process\ time$, $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$, which have been defined in Section 3.2.2. Our solution has to fulfill these requirements, too, for being integrable into the process chain of AM service providers.

9.2.1 Recognition Rate

In our experiments for the evaluation of the performance of our solution regarding the requirement $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$ for the two data sets Random30 and Similar50, we are dealing with the influence of the following factors: At first, we compare raw training images generated using random orientations, augmented training images generated using random orientations, raw training images generated using physically sound orientations and physically sound training images with additional augmentation. For all of these four cases, we used training data sets with 60 respectively 240 different training instances per component and compared the two variants v_1 and v_2 for the virtual camera array settings. Beside that factors, we evaluated the influence of the different augmentation techniques on the physical inference images. By augmenting the inference images, we can on the one hand draw conclusions about the robustness of our solution with respect to changes in the inference image generation and on the other hand verify whether an augmentation of the inference images leads to an improvement of the recognition rate. The results are presented in the following tables.

Random30

We start with the results on the Random30 data set. Table 9.1 shows the results for the Random30 data set, using the randomly generated training data. The first six rows are showing the results using 60 training instances per object. It can clearly be seen that the RotationNet instances, which have been trained on augmented training data, outperform the RotationNet instances which have been trained on raw data. Using raw training data and a fixed angle of 45 degree to the horizontal plane, the best result of 77.61% accuracy is achieved using sharpened physical images is. Using the composition of augmentation techniques, an accuracy of 94.03% can be achieved on the raw physical images. Additionally, it can be seen that the camera array variant v_2 which is using the alternating angle of 35 respectively 55 degree to the horizontal plane, leads to a much better result of 89.55% accuracy for the raw data, since it provides additional information due to the higher variations

Table 9.1 Evaluation of the performance of RotationNet on the Random30 data set using randomly generated training data with and without augmentation.

Random30 - Random Orientations						
	Camera Array	Inst.	Raw	RBC	Sharpening	Sharpening & CLAHE
Raw	v_1	60	68.66%	73.13%	**77.61%**	68.66%
RBC	v_1	60	86.57%	88.06%	**91.04%**	88.06%
Composition	v_1	60	**94.03%**	91.04%	91.04%	91.04%
Raw	v_2	60	86.57%	83.58%	**89.55%**	88.06%
RBC	v_2	60	85.57%	88.06%	**91.04%**	86.57%
Composition	v_2	60	**95.52%**	92.54%	92.53%	91.04%
Raw	v_1	240	76.12%	76.12%	83.58%	**88.06%**
RBC	v_1	240	**97.01%**	**97.01%**	94.03%	94.03%
Composition	v_1	240	**97.01%**	95.52%	**97.01%**	95.52%
Raw	v_2	240	80.60%	85.07%	**91.04%**	85.07%
RBC	v_2	240	89.55%	91.04%	**97.01%**	92.54%
Composition	v_2	240	**97.01%**	**97.01%**	**97.01%**	95.52%
Mix	v_1, v_2	240	**97.01%**	**97.01%**	**97.01%**	95.52%

between the different viewpoints. Using the camera array variant v_2, an increased accuracy of 95.52% can also be achieved for the composition of augmentation techniques.

A line-by-line comparison of the augmentation techniques for the physical inference images reveals another effect. When using raw data or RBC, especially the sharpening filter for the physical images leads to an improvement of the recognition rate. In contrast, for the RotationNet instances trained using the data augmented by the composition of augmentations, the augmentation of the physical inference data has a slightly negative effect on the recognition rate of the system. Nevertheless, the instance trained using the composition of augmentations is overall most robust to variations in the inference data. By randomly combining the different augmentation techniques, the system is optimally trained to recognize the physical inference images.

The following seven rows are showing the results for the RotationNet instances trained on 240 training instances per object. The trends between raw data and augmented data already described, can be confirmed here. Overall, quadrupling the training data further improves the recognition rate of the system. Especially for the raw data, the accuracy increases from 68.66% to 88.06%. Both, the instances trained using RBC and the composition can achieve recognition rates up to 97.01%.

Table 9.2 Evaluation of the performance of RotationNet on the Random30 data set using physically sound training data with and without augmentation.

Random30 - Phyiscally Sound Orientations						
	Camera Array	Inst.	Raw	RBC	Sharpening	Sharpening & CLAHE
Raw	v_1	60	**98.51%**	97.01%	92.54%	83.58%
RBC	v_1	60	94.03%	92.54%	**97.01%**	94.03%
Composition	v_1	60	**95.52%**	91.04%	91.04%	83.58%
Raw	v_2	60	**98.51%**	97.01%	97.01%	94.03%
RBC	v_2	60	**97.01%**	**97.01%**	**97.01%**	**97.01%**
Composition	v_2	60	**98.51%**	**98.51%**	97.01%	91.04%
Raw	v_1	240	**95.52%**	92.54%	91.04%	86.57%
RBC	v_1	240	97.01%	92.54%	**98.51%**	92.54%
Composition	v_1	240	**98.51%**	**98.51%**	**98.51%**	95.52%
Raw	v_2	240	**98.51%**	92.54%	92.54%	91.04%
RBC	v_2	240	95.52%	94.03%	**98.51%**	95.52%
Composition	v_2	240	**98.51%**	**98.51%**	97.01%	97.01%
Mix	v_1, v_2	240	**97.01%**	**97.01%**	**97.01%**	95.52%

As a final additional experiment for random training data on the Random30 data set, we generated a mixed training data set. This training set consists of $1/4$ raw images with fixed angles, $1/4$ augmented images using RBC and fixed angles, $1/4$ augmented images using the composition of augmentations and fixed angles as well as $1/4$ using the composition and alternating angles.

Similar to the instance of RotationNet, which has been trained using the data augmented by the composition of different techniques, using the mixture of the different sets, a stable recognition rate of 97.01% can be achieved for nearly all of the augmentation techniques used for the physical inference data. Thus, it can be stated that a combination of different augmentation techniques leads to a higher variation in the training data, making the system trained on synthetic data generalize better to the physical data and enables it to transfer the performance from synthetic to physical data.

Table 9.2 is showing the results for the Random30 data set and physically sound training data. It can be seen that there are significantly fewer differences between the performances of the individual variants. In all cases, the trained instances achieve at least 95.52% accuracy on at least one of the physical inference data sets, regardless of the augmentation technique used for training. The maximum achievable accuracy is 98.51%. It is reached for multiple training data variants for both cases, using 60 or 240 training data instances per 3D model. The worst

Table 9.3 Random vs. physically sound training data with different augmentations. The trained models were evaluated on photos of physical objects of the Random30 data set.

Random30 - Phyiscally Sound vs. Random Orientations				
	Camera Array	Inst.	Random	Physically Sound
Raw	v_1	60	77.61%	**98.51%**
RBC	v_1	60	91.04%	**97.01%**
Composition	v_1	60	94.03%	**95.52%**
Raw	v_2	60	89.55%	**98.51%**
RBC	v_2	60	91.04%	**97.01%**
Composition	v_2	60	95.52%	**98.51%**
Raw	v_1	240	88.06%	**95.52%**
RBC	v_1	240	97.01%	**98.51%**
Composition	v_1	240	97.01%	**98.51%**
Raw	v_2	240	91.04%	**98.51%**
RBC	v_2	240	97.01%	**98.51%**
Composition	v_2	240	97.01%	**98.51%**
Mix	v_1, v_2	240	97.01%	**98.51%**

case arises for the RotationNet instance trained on 60 raw training data instances per object and the evaluation on the physical images augmented with sharpening and CLAHE. In this case, a recognition rate of 83.58% is achieved.

Compared to that, for the randomly generated training data, the different instances have at least achieved 77.61% accuracy on one of the physical inference data sets and had a worst case of 68.66% for the case of using 60 raw training data instances per object and again the evaluation on the physical images augmented with sharpening and CLAHE.

Table 9.3 shows the best results of both the randomly generated and the physically sound training data for comparison. These results show that using physically sound training data can achieve a performance increase to a maximum of 98.51% in comparison to 97.01% for the randomly generated training data even with standard augmentation techniques. Since we first examined the simplest data set Random30, already the RotationNet instances trained on randomized data are able to achieve high recognition rates close to 100% when using the composition of different augmentations. The main difference for the Random30 data set is that the RotationNet instances trained on physically sound data show less variation in performance with respect to changes in the inference data and are therefore more robust to variations in the physical image generation step.

Table 9.4 Evaluation of the performance of RotationNet on the Similar50 data set using randomly generated training data with and without augmentation.

Similar50 - Random Orientations						
	Camera Array	Inst.	Raw	RBC	Sharpening	Sharpening & CLAHE
Raw	v_1	60	15.04%	18.05%	19.55%	**37.59%**
RBC	v_1	60	57.14%	57.14%	**61.65%**	58.65%
Composition	v_1	60	60.90%	59.40%	60.90%	**62.14%**
Raw	v_2	60	45.86%	44.36%	52.63%	**53.38%**
RBC	v_2	60	58.65%	60.15%	**66.17%**	57.14%
Composition	v_2	60	**69.92%**	66.92%	63.16%	66.92%
Raw	v_1	240	38.35%	42.86%	45.11%	**47.39%**
RBC	v_1	240	51.13%	60.15%	**63.91%**	57.14%
Composition	v_1	240	66.17%	**69.17%**	67.67%	**69.17%**
Raw	v_2	240	38.25%	39.98%	45.11%	**50.38%**
RBC	v_2	240	54.89%	56.39%	**69.92%**	60.90%
Composition	v_2	240	67.67%	66.92%	**69.92%**	60.90%
Mix	v_1, v_2	240	**71.43%**	**71.43%**	**71.43%**	70.68%

Similar50

The tests for the Random30 data set have shown that our solution is basically able to achieve a very high recognition rate for the recognition of additively manufactured components. With another experiment based on the Similar50 data set, we want to evaluate the performance of our approach for very similar components, which are extremely difficult to distinguish.

Again, we begin with the results generated using randomly generated orientations for the training data generation. As can be seen in Table 9.4, the accuracies for the Similar50 data set are significantly lower, compared to the Random30 data set. The RotationNet instances trained using raw data without augmentation for the training or inference data achieve a maximum of 45.86% recognition rate. Using the augmentation techniques sharpening and CLAHE on the physical inference images, the accuracy can be increased to 53.38% recognition rate.

The performance can be enhanced by using augmented training data again. Similar to the results for the Random30 data set, the random composition of different augmentation techniques leads to the best results for both camera array variants, v_1, the fixed camera angle of 45 degree from the horizontal plane and v_2, the alternating angles of 35 and 55 degree. For both cases, a recognition rate up to 69.92% can be achieved what is significantly better compared to raw training data.

Table 9.5 Evaluation of the performance of RotationNet on the Similar50 data set using physically sound training data with and without augmentation.

Similar50 - Phyiscally Sound Orientations						
	Camera Array	Inst.	Raw	RBC	Sharpening	Sharpening & CLAHE
Raw	v_1	60	74.44%	72.18%	71.43%	**75.94%**
RBC	v_1	60	79.70%	**80.45%**	79.70%	77.44%
Composition	v_1	60	80.45%	**81.95%**	78.20%	78.95%
Raw	v_2	60	69.92%	69.17%	68.42%	**71.43%**
RBC	v_2	60	73.68%	**77.44%**	**77.44%**	75.19%
Composition	v_2	60	**79.70%**	78.95%	78.95%	77.44%
Raw	v_1	240	61.65%	61.65%	60.90%	**66.17%**
RBC	v_1	240	**81.95%**	78.20%	78.20%	76.69%
Composition	v_1	240	79.70%	78.20%	79.70%	**80.45%**
Raw	v_2	240	66.17%	66.17%	63.16%	**67.67%**
RBC	v_2	240	75.94%	71.43%	**76.70%**	70.68%
Composition	v_2	240	**82.71%**	81.95%	81.95%	81.95%
Mix	v_1, v_2	240	81.95%	82.71%	**84.21%**	81.95%

Again, we created an additional training data set, by mixing the data sets of $1/4$ raw images with fixed angles, $1/4$ augmented images using RBC and fixed angles, $1/4$ augmented images using the composition of augmentations and fixed angles as well as $1/4$ using the composition and alternating angles. Using that training data, the system can achieve up to 71.43% recognition rate what is the best value for the randomly generated training data. This result is again showing that the highest possible variation in the training data and the use of additional information through the two different camera positioning variants leads to the best results.

In summary, the achievable recognition rate for the difficult Similar50 data set using randomly generated training data is significantly lower than for the Random30 data set and is far from hundred percent recognition rate.

Table 9.5 is showing the results for the Similar50 data set using physically sound training data. It can clearly be seen that the RotationNet instances trained using physically sound training data outperform the instances trained using randomly generated training data. Using raw data, recognition rates up to 75.94% can be achieved which is already higher than the best case recognition rate of 71.43% for the randomly generated data. Again, this recognition rate can be increased up to 82.71% using 240 instances of images generated using the variable angles and augmented using the composition of augmentation techniques. Similar to the

Table 9.6 Random vs. physically sound training data with different augmentations. The trained models were evaluated on photos of physical objects of the Similar50 data set.

Similar50 - Phyiscally Sound vs. Random Orientations				
	Camera Array	Inst.	Random	Physically Sound
Raw	v_1	60	37.59%	**75.94%**
RBC	v_1	60	61.65%	**80.45%**
Composition	v_1	60	62.14%	**81.95%**
Raw	v_2	60	53.38%	**71.43%**
RBC	v_2	60	66.17%	**77.44%**
Composition	v_2	60	69.92%	**79.70%**
Raw	v_1	240	47.39%	**66.17%**
RBC	v_1	240	63.91%	**81.95%**
Composition	v_1	240	69.17%	**80.45%**
Raw	v_2	240	50.38%	**67.67%**
RBC	v_2	240	69.92%	**76.70%**
Composition	v_2	240	69.92%	**82.71%**
Mix	v_1, v_2	240	71.43%	**84.21%**

case of randomly generated training data, this value can even be increased up to 84.21% by mixing the different training data sets what especially includes a mix of training data generated using the two different camera array variants v_1 and v_2. This variation seems to have a major positive impact, since it provides additional information.

An interesting effect is that the recognition rate drops when using raw training data and 240 instances. This seems to occur due to the data generation method, which we used. If too many very similar training data instances are generated from the few physically sound orientations, the model overfits and is no longer able to transfer its performance to the physical inference data. By using the augmentation techniques, this effect can be avoided, since more variation is generated within the training data set.

In Table 9.6, the results of the RotationNet instances trained using randomly generated orientations are compared with the instances trained on physically sound training data. The direct comparison shows that the RotationNet instances trained on physically sound training data outperform the instances trained on randomly generated training data in all cases.

These results demonstrate that our augmentation technique which uses the geometric properties of the components to be recognized for an adaptive augmentation of the training data, clearly outperforms common augmentation methods. By generating the training data in a physically sound way, the distribution of the training data can be adapted to the distribution

of the physical inference data and thus, the transfer between virtual and physical domain can be optimized.

The results show a strong correlation between the complexity of the data set and the recognition rate of our system. For a relatively simple data set with 30 randomly chosen components, recognition rates very close to 100% can be achieved. However, for more and especially more similar parts, the recognition rate decreases. As already mentioned in Section 9.1.1, a data set like Similar50 does not actually occur in real production. However, in order to be able to guarantee a very high recognition rate, independent of the composition of a build job, we have made further assumptions for a future solution, which we will explain in Section 9.3.

In addition to the observation of the maximum recognition rates, which we have already described above, it is also of interest to analyze the development of the performance of our solution over the training time. We consider this using the example of the Similar50 data set.

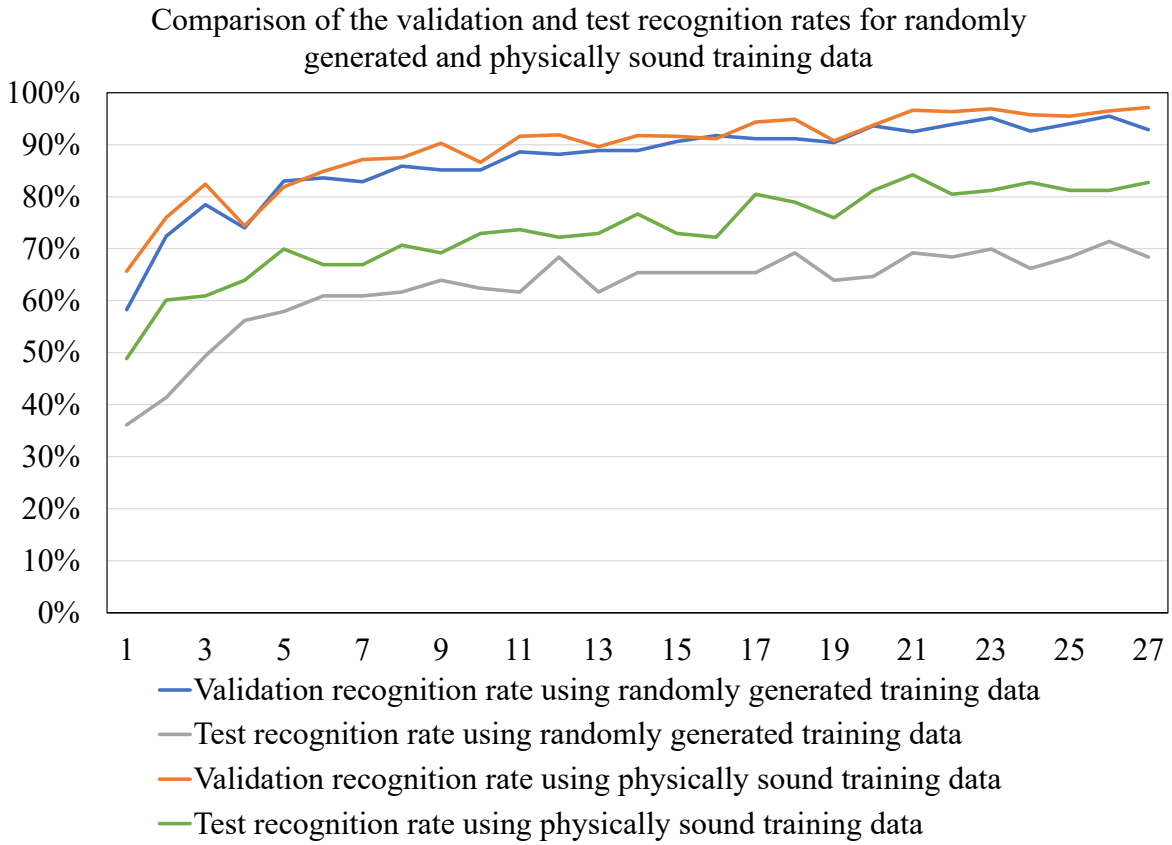


Fig. 9.9 The graph shows the validation and test recognition rates for randomly generated and physically sound training data over the training iterations in thousands.

Figure 9.9 shows the development of the validation and test accuracy for an instance of RotationNet which has been trained using randomly generated training data and an other instance trained using physically sound data. As can be seen, the courses of the two validation recognition rates do not differ a lot. The reason for this is that the validation data sets each originate from the associated distribution of the corresponding training data set and are therefore not the same. In contrast, there are clear differences in the progressions of the recognition rates on the test data. The recognition rate of the system which has been trained with physically sound training data, is consistently five to 15 percent higher and reaches its maximum of 84.21% at iteration 21000. The maximum recognition rate of 71.43% for the randomly generated training data is reached at iteration 26000.

For the evaluation of the recognition rate based on the physical test images, we used the RotationNet state with the highest recognition rate for the validation data set in each of the results described above. This state does not necessarily provide the highest recognition rate for the physical data. However, as can be seen in Figure 9.9, the recognition rate for the physical data settles at a very high level.

Another important aspect is the influence of using the RotationNet instance pre-trained on the ILSVRC data set and subsequent transfer learning of the model. Figure 9.10 is showing the courses of the validation recognition rates for RotationNet instances trained on randomly generated and physically sound training data with and without pre-training on the Similar50 data set and Figure 9.11 is showing the corresponding test recognition rates. As can be seen in Figure 9.10, using the pre-trained instance of RotationNet leads to validation recognition rates around 60% after 1000 training iterations for both, randomly generated and physically sound training data. In comparison, the instance trained on physically sound training data without pre-training needs about 6000 training iterations to reach that recognition rate and the instance trained using randomly generated training data even needs around 10000 Iterations. Additionally, the RotationNet instances trained without pre-training can only achieve a maximum of 90% validation recognition rate compared to 95.5% respectively 97% for the pre-trained instances.

The same can be seen in Figure 9.11, which is showing the corresponding test recognition rates. In both cases, for the randomly generated and physically sound training data, the maximum recognition rate is about 15% higher when pre-training is used. These results show the high influence of pre-training and subsequent transfer learning. Pre-training on the ILSVRC data set contributes a large amount of additional information that adds extreme value for solving the recognition problem. Without pre-training and transfer learning, the performance of our solution system would be significantly lower. The results show that

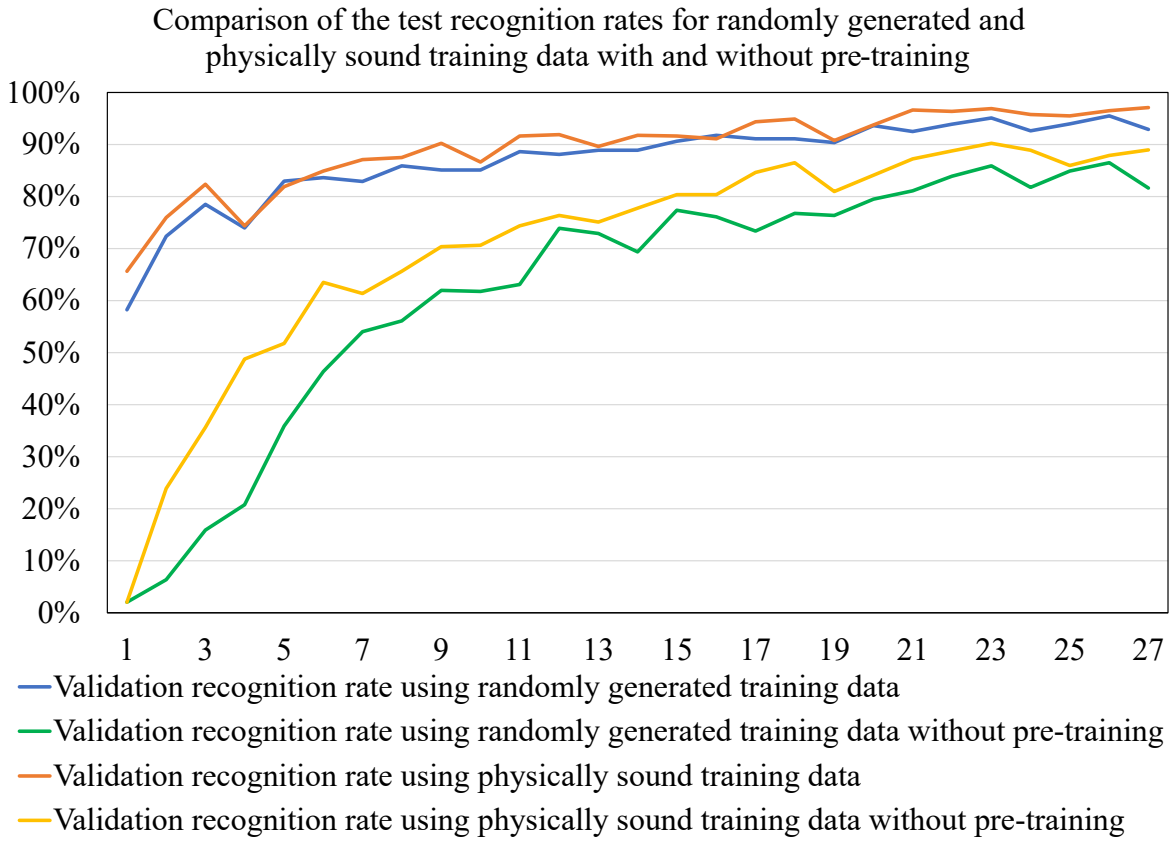


Fig. 9.10 The graph shows the validation recognition rate for randomly generated and physically sound training data with and without pre-training over the training iterations in thousands.

the use of the transfer learning technique can significantly improve the performance of DL models when dealing with problems for which only a small amount of data is available.

The experiments described above have shown that our solution is capable to recognize additively manufactured components with a high recognition rate. For standard build jobs as they mainly occur in the daily production of AM service providers, we achieve a very good recognition rate. However, as the recognition rate in the current state of our solution is still dependent on the complexity of the build jobs, we rate the performance of our solution with respect to the requirement $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$ as good. In section 9.3, we will discuss how the recognition rate can be further increased even for complex build jobs. Before we address this, we will deal with the remaining requirements.

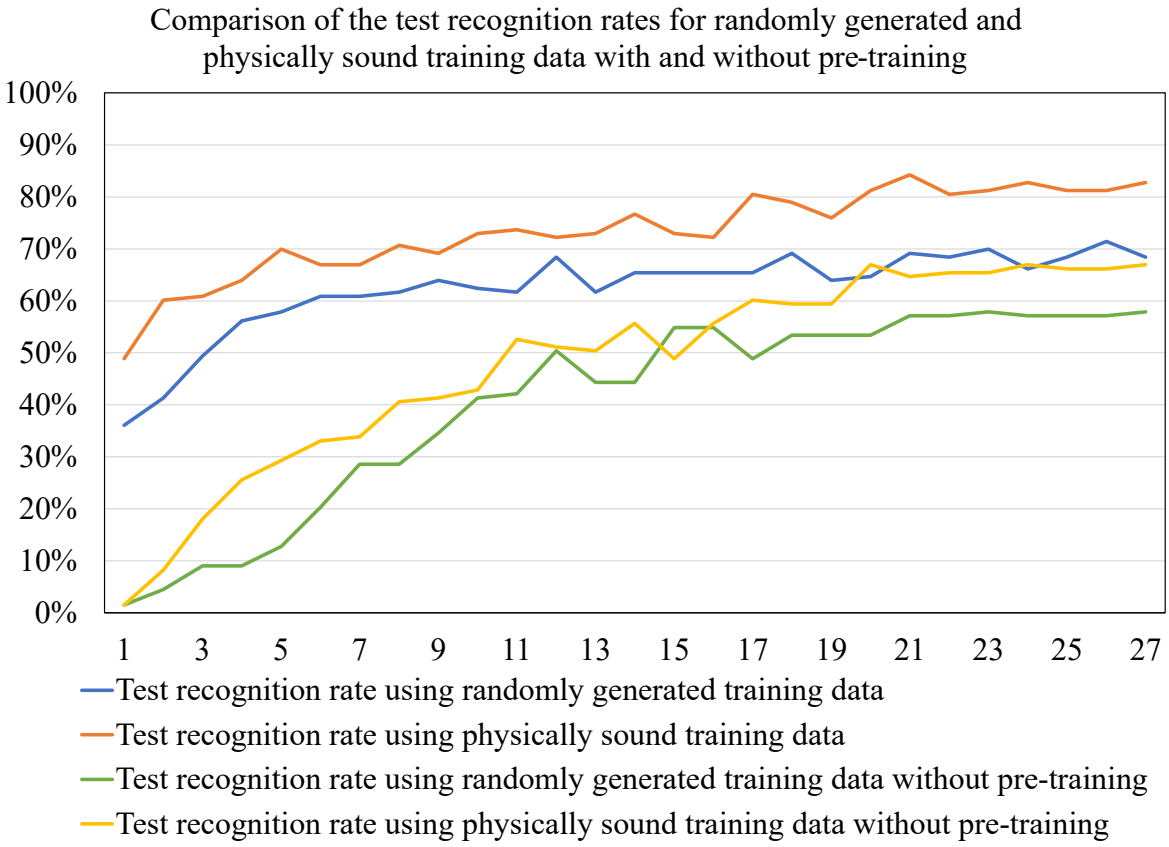


Fig. 9.11 The graph shows the test recognition rate for randomly generated and physically sound training data with and without pre-training over the training iterations in thousands.

9.2.2 Secondary Requirements

As explained in Section 3.2.2, besides the requirement $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$ the additional requirements $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{3-cr} \mathrel{\widehat{=}} adaptability$, $R_{4-cr} \mathrel{\widehat{=}} process\ time$, $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$ have to be fulfilled.

$R_{2-cr} \mathrel{\widehat{=}} robustness$ implies that the recognition rate should be independent of changes, e.g. in the creation of physical images due to, for example, contamination or abrasion in the scanning area. The results described in Section 9.2.1, have shown that an appropriate augmentation of the training images using a composition of photometric image augmentation techniques, can ensure a high recognition rate, regardless of e.g. changes in the lighting condition or sharpness of the inference images. Especially the random composition of the techniques Gaussian noise, RBC, blurring and CLAHE has led to a high recognition rate for all test series, regardless of the changes in the inference images. We therefore rate the quality of our solution regarding the requirement $R_{2-cr} \mathrel{\widehat{=}} robustness$ as excellent.

By using a data-driven approach, the requirement $R_{3-cr} \mathrel{\widehat{=}} adaptability$ is fully met by our solution. By training the individual instances of RotationNet for each build job on a daily basis, the system learns which criteria are decisive for the recognizing of the components corresponding to a build job, based on the corresponding set of 3D models. As a result, the features with the highest discriminative information are learned for each build job. The necessary steps for the generation of physically sound training data and the training itself are executed completely automated. Thus, no manual adjustments are necessary for a specific build job. We therefore rate the performance of our solution with regard to the criterion R_{3-cr} $\mathrel{\widehat{=}} adaptability$ as excellent.

Regarding the requirement $R_{4-cr} \mathrel{\widehat{=}} process\ time$, two aspects have to be considered. As explained in Section 3.2.2, our solution has to be able to adapt to the daily changing production within about 21 hours and subsequently recognize up to a few thousand different components within two to four hours.

For our solution, the adaptation to the daily changing production corresponds to the generation of training data and the training of a RotationNet instance for each build job. Since the training of multiple instances can be parallelized, we consider the time for the generation of training data and training of a single instance of RotationNet in the following. For the training data generation, at first the physically sound orientations have to calculated for each 3D model of a build job. Depending on the complexity of the 3D models, this takes between a few seconds up to a few minutes. The generation of the training and validation images itself takes between 5 and 10 minutes per 3D models when using 240 training and 30 validation instances consisting of 12 images each on a common notebook. The generation of the images for the different 3D models can be parallelized. The augmentation of the training and validation images takes about 2 minutes per 3D model and can be parallelized, too.

Since the difficulty of the Random30 and Similar50 data set differs from each other, the necessary training time itself differs quite a lot. The Figures 9.12 and 9.13 show the development of the recognition rate for the physical inference images as a function of training time for the instances trained using the mix of different augmentation techniques as described in Section 9.2.1. As can be seen in Figure 9.12, for the Random30 data set, the recognition rate of 98.51% can be achieved in about 100 minutes. After 80 minutes, a recognition rate of 97.51% can already be achieved. The instance trained on randomly generated data reaches its best recognition rate of 97.01% after about 120 minutes. For this data set, which represents an average build job occurring in reality, it is therefore easily possible for our solution to learn the recognition of the components of a build job in the time available.

As can be seen in Figure 9.13, the training time for the Similar50 data set is much longer. For the RotationNet instance trained on physically sound training data, the highest

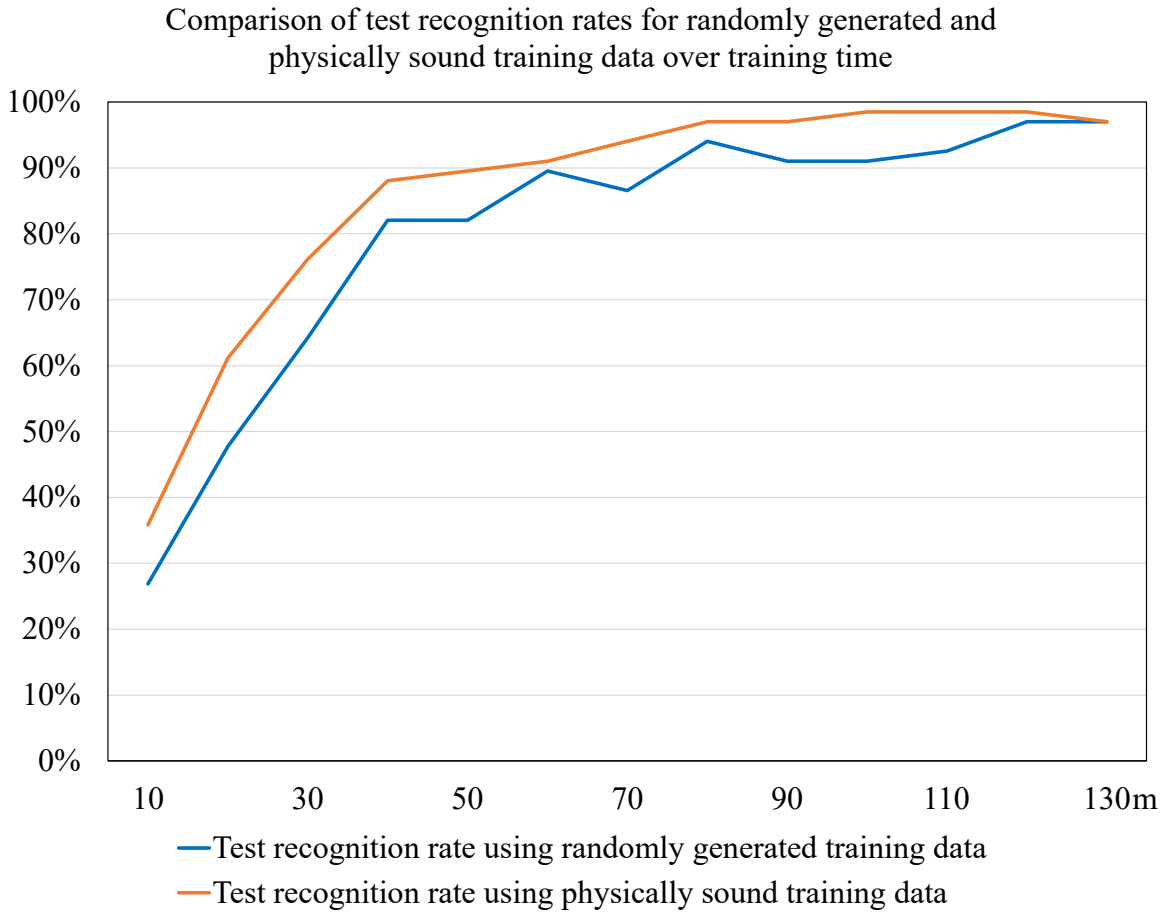


Fig. 9.12 The graph shows the recognition rate on the physical inference data of the two instances trained on randomly generated and physically sound training data in percent over the training time in minutes for the Random30 data set.

recognition rate is reached after about 33 hours of training. For the instance trained using randomly generated training data, it takes about 41 hours. That is significantly longer than for the Random30 data set and beyond the 21 hours available for the training, parallel to the physical process chain of AM service providers. In the following Section 9.3, we are providing a concept for an adaption of our solution, which guarantees that the training time which is necessary for achieving a high recognition rate, is well below the existing time constraints of 21 hours in all cases.

Besides the requirement regarding the training time, we also investigated the time necessary for the recognition of a produced component itself. With a batch size of 1, which is used for the inference phase, RotationNet is able to assign a component in 0,81 seconds on average based on the 12 images of the component. Therefore, it is capable to recognize about 4400 components per hour. Of course, this time span only describes the pure computation

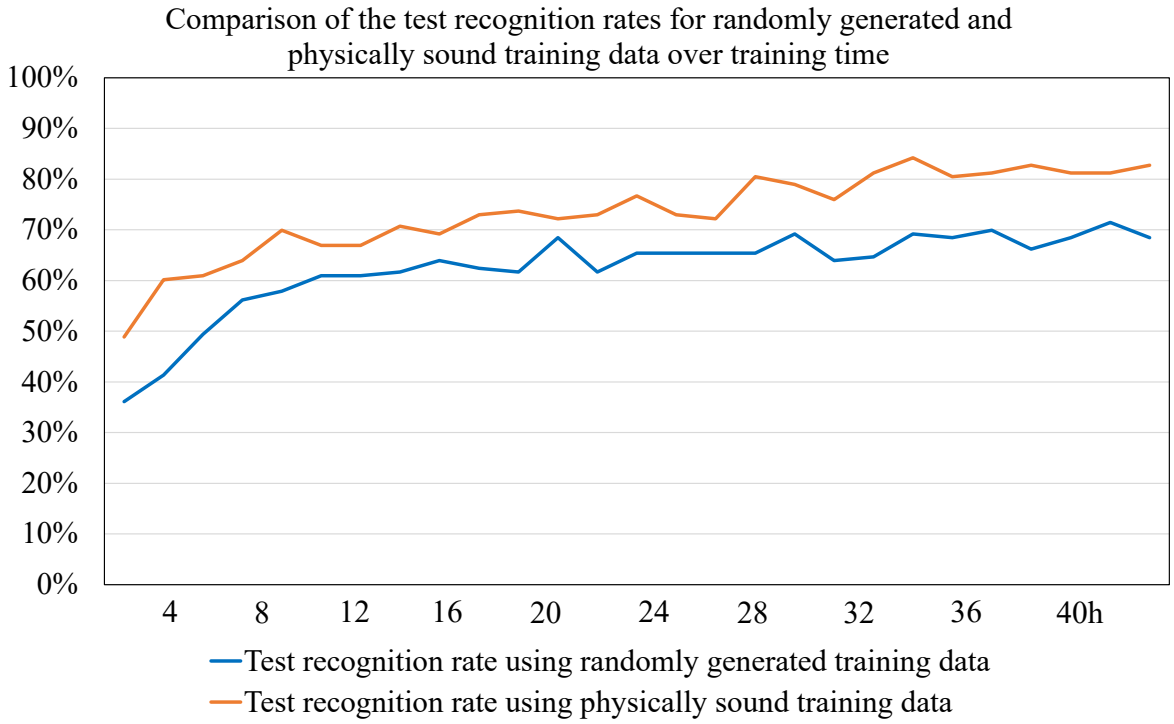


Fig. 9.13 The graph shows the recognition rate on the physical inference data of the two instances trained on randomly generated and physically sound training data in percent over the training time in hours for the Similar50 data set.

time for processing the images. The time to recognize a component is mainly influenced by the time it takes to move a single component on the conveyor belt into the scanning area and generate the 12 images. Since the physical recognition station does not yet exist, we currently can not evaluate the time required for recognizing a single component. If it supposed to be possible to recognize 1000 components per hour, the mechanical process should take a maximum of 3.6 seconds. Based on the described calculations, we evaluate the performance of our solution with respect to the $R_{4-cr} \mathrel{\widehat{=}} \textit{process time}$ requirement as good.

The quality of our system with regard to the requirement $R_{5-cr} \mathrel{\widehat{=}} \textit{scalability}$ can also be rated as good. Standard hardware is sufficient for the computational effort for the creation of the training data, the subsequent training and finally the inference phase. Therefore, as production numbers increase, the hardware can be adapted accordingly if necessary. There could be a limitation due to the design of the recognition station, as it can only move the individual components into and out of the scanning area at a limited speed. With significantly increasing production numbers, this setup may reach its limits, which would require a second identical setup.

Table 9.7 Comparison of our solution with the manual process and the general classes of traditional and DL-based CV approaches regarding the six requirements recognition rate, robustness, adaptability, process time, scalability, and process costs.

Method/Metric	R_{1-cr} Recognition Rate	R_{2-cr} Robustness	R_{3-cr} Adaptability	R_{4-cr} Process Time	R_{5-cr} Scalability	R_{6-cr} Process Costs
Manual process	++	++	+	−−	−	−−
Traditional CV approaches	++	++	−	−−	+	+
DL-based CV approaches	++	++	++	+	+	+
Our Solution	+	++	++	+	+	+

This would also influence the costs for our solution. The main expense is the hardware for the conveyor belt and the scanning area. Adding a second system for increasing production numbers, would double the initial costs for the system. However, by using a 2D data-based approach, the much cheaper cameras compared to 3D scanners can be used. Since common GPUs are sufficient for the training of the RotationNet architecture, these are not a major cost driver. Therefore, we also rate the quality of our solution with regard to the requirement $R_{6-cr} \mathrel{\widehat{=}} \textit{process costs}$ as good.

In Table 9.7, we compare the performance of our solution with the manual approach and our assumptions regarding traditional and DL-based CV approaches. As can be seen, our solution nearly achieves the performance which we have predicted for DL-based solutions. Only the accuracy is lacking a bit for very complex build jobs. Nevertheless, as stated in Section 9.2.1, we already developed a concept for an adaption of our solution, which enables it to achieve accuracies close to 100% also for complex build jobs. The concept will be explained in Section 9.3.

Additionally, it can be seen that our solution outperforms the current manual process with respect the requirements$R_{4-cr} \mathrel{\widehat{=}} \textit{process time}$, $R_{5-cr} \mathrel{\widehat{=}} \textit{scalability}$ and $R_{6-cr} \mathrel{\widehat{=}} \textit{process costs}$. Our solution is therefore able to recognize produced components with a high accuracy and robustness, saving time and costs compared to the manual process. In the next section, we will discuss which problems currently still exist and how they can be solved.

9.3 Discussion

Our experiments have shown that our solution based on RotationNet and physically sound training data is able to recognize additively manufactured components with a high recognition rate, based on 12 images taken per component. Using the approach of physically sound training data, enhances the recognition rate of our system by aligning the training data with the physical inference data. The system is able to automatically adapt for the recognition of the daily produced components by training individual instances of RotationNet for each build job. For this purpose, an instance pre-trained on the ILSVRC data set can be used to further increase the recognition rate of the system using transfer learning while reducing the training time.

With our experiments, we have shown that the recognition rate strongly depends on the size of the data set and the similarity of the associated components. While the recognition rate for a common build job, like the Random30 data set, is very close to 100%, only about 84% accuracy can be achieved for the extremely complex data set Similar50. Also, the training time increases for the complex Similar50 data set to a period beyond the available 21 hours.

This leads us to further considerations to be able to achieve both a high recognition rate and a training time within the 21 hour interval. The clear correlation of the complexity of a data set with the recognition rate and required training time suggest a simplification of the data sets. Currently, we simply use all 3D models belonging to a build job together and train the MVCNN-based system to recognize the components based on visual features only. However, in addition to visual recognition, other properties can be used to allow pre-sorting and thus simplify the complexity of the problem to be solved for our visual recognition system. This requires information based on which both the digital 3D models and the produced physical components can be pre-sorted. Two possibilities to implement pre-sorting are created by using the size or the weight of the components.

On the digital side, the size can easily be extracted from the information of the 3D models. The physical components can also be sorted, e.g. into components larger and smaller than a certain threshold value, using a screening machine. In this way, the build job data sets could be split into two or more sub-sets. According to the divide and conquer principle, the components of the sub-sets are easier to recognize than the components of the whole data set used as a single set. For each sub-set, a separate RotationNet instance could be trained for the associated components and subsequently used for the inference.

By using the weight of the individual components for the division of the data sets, they could be divided even more fine-grained. For the digital 3D models, their volume and the density of the material used for production can be used to calculate the expected weight of

the components to be produced. The weight of the physical components can be determined by a weighing that can be integrated into the conveyor belt. The weight of the components can vary due to powder adhesions or residues in complex geometries and thus deviate from the weight calculated on the basis of the 3D models. Therefore, when subdividing the data sets with the help of the weight criterion, categories with a certain margin would have to be created.

With the help of the two criteria or the combination of both, the recognition tasks for each build job could be divided and simplified and thus the recognition rate of our solution could be further increased. Even more important is the effect, that the training time drastically decreases for simpler data sets. Using the approach of dividing the build job data sets into sub-sets, we can guarantee that the training time does not exceed the available 21 hours.

Another factor that affects the recognition rate is the image generation of the physical inference data. As we have described in Chapter 7, it is extremely important to align the synthetically generated training data and the physically generated inference data in an optimal way to ensure a high recognition rate. Therefore, we adapted the synthetic training data generation process to the conceptually created physical recognition station. However, since the recognition station does not yet exist, we manually created the images for our experiments. This leads to several problems. First, without professional lighting systems, the lighting could not be completely adapted to the lighting of the virtually defined scene. This leads to discrepancies between the shadows which are cast for the synthetic training data and the physical inference data, thus affecting the recognition of the components. On the other hand, due to the manually guided camera, the camera angle of 45 degrees to the horizontal plane and the 30 degree steps around the Z-axis between the individual viewing angles could not be guaranteed completely accurate. With a fixed installation of the cameras and the lighting, these two factors could be adapted to the synthetic data with a higher accuracy, which would presumably further optimize the recognition rate.

9.4 Summary

We have shown that it is possible to develop a solution system, which is able to recognize additively manufactured components with a high recognition rate and robustness, based on a state-of-the-art 3D DL approach. Additionally, the system is able to automatically adapt to the daily changing production. Despite the lack of physical data, the system can be trained with synthetically generated data, which has been adapted to the physical conditions, in order to recognize produced components on the basis of real images.

The system has been designed in a way that all necessary data processing steps from training data generation to inference data generation are performed in a fully automated manner, without the need for manual intervention. The necessary information is extracted automatically from the data process chain of the AM service provider, allowing the process steps required for the *3D Component Recognition* to be run in parallel with the physical production of the components to be recognized.

The experiments conducted so far have not yet been carried out in the actual industrial environment. As already described in Section 9.3, we believe that our solution system can be further optimized by an extended use of the existing process information. By integrating the described solution system into the process chain of AM service providers, the share of manual work in the overall process can be reduced, resulting in time and cost savings.

Chapter 10

Conclusion and Future Work

In this chapter, we finally want to summarize the central aspects of our work. Since our work is dealing with the two sub-processes of *Manufacturability Analysis* in the process chain of AM service providers and *3D Component Recognition* of additively manufactured components, we already presented a summary regarding our work on these two process steps in the corresponding Chapters 8 and 9. Nevertheless, in this chapter, we are going to summarize our main findings regarding the possibilities of automating the process chain of AM service providers with state-of-the-art DL methods.

10.1 Conclusion

At the beginning of this thesis, in Chapter 1, we have outlined that AM is still a relatively young and growing manufacturing technique, which has to contend with complex and dynamic issues as part of its growth. Especially, the business model of AM service providers requires flexible solutions for automating the associated complex process chain, in order to be able to process the further increasing production volumes. The goal of our work was to show that innovative data-driven solutions are capable of being used for the further automation of this strongly data-driven process chain.

In the context of this thesis, we therefore first analyzed the process chain of AM service providers from the beginning of the ordering process to the final delivery of the ordered components in Chapter 2. The two sub-processes of *Manufacturability Analysis* for AM and *3D Component Recognition* for additively manufactured components were chosen as central aspects for our work, since they offer high potentials for further optimization through automation. Based on the analysis of the current approaches of the two process steps, existing issues were identified and requirements for solutions to automate the process steps were defined in Chapter 3. In order to be able to develop a solution optimized with regard to the

requirements for each of the two process steps, we have identified state-of-the-art methods of ML, DL and especially 3D-DL in Chapter 4 and provided an overview of alternative solutions for the two process steps in Chapter 5.

Based on this information, we developed the solutions which we presented in Chapter 6 and 7. In this section, we will present a conclusion on how the two solutions can be integrated into the process chain of AM service providers. For this purpose, we will address the requirements defined in the Sections 3.1.3 and 3.2.2 and discuss the extent to which they have been met.

10.1.1 Manufacturability Analysis

For the process step *Manufacturability Analysis*, we have defined the requirements $R_{1-ma} \mathrel{\widehat{=}} adaptability$, $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ and $R_{5-ma} \mathrel{\widehat{=}} computation\ time$. The first four requirements have to be met to provide the general functionality of the *Manufacturability Analysis*, the last requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ is relevant to enable the solution to be integrable into the process chain of AM service providers.

We have designed a 3D-CNN-based solution, which is able to learn to detect critical geometries within 3D models, based on a labeled data set and furthermore visualize the detected geometries using the LRP approach. The solution is designed for being able to learn which geometric properties are decisive for the manufacturability of components based on production data which is collected and labeled within the daily production processes of AM service providers. To evaluate the performance of the two approaches, we conducted the experiments described in Chapter 8, which are designed to reflect the real-world conditions at AM service providers as accurate as possible.

As described in Chapter 8, we used the criterion of *minimum wall thickness* as an example for the *Manufacturability Analysis*. Since not enough production data has been collected at the current time and no benchmark data sets exist yet, we designed our own data set for this purpose. The conducted experiments have show that our solution based on the HCNN architecture is able to correctly classify the 3D models used as test data for our experiments into the two classes manufacturable and non-manufacturable with an accuracy of about 84%. This accuracy is reached using a training data set consisting of 1200 critical and 1200 non-critical geometries. With our experiments, we have also shown that the *minimum wall thickness* criterion can be detected with an accuracy of over 80% when about 400 3D models which contain critical geometries are available for the training. This information can be used to estimate how many 3D models need to be collected in the context of real production for learning further criteria. Besides the pure classification of the 3D models, we have shown

that our solution can also be used for the visualization of critical geometries. With the help of the LRP approach, the most relevant areas of the geometry for the decision of the HCNN model, can be highlighted. On the one hand, this allowed us to verify that the correct features were learned even though the training data contained only the purely geometric information, and on the other hand, the visualization can be used to generate further information for the customers of AM service providers, which these can use for the revision of their 3D models if necessary.

Our solution has shown that it is able to evaluate the manufacturability of 3D models with respect to the manufacturability criterion *minimum wall thickness* with a good accuracy and additionally generate a high resolution visual feedback. Thus, we have shown that the main requirements $R_{2-ma} \mathrel{\widehat{=}} accuracy$, $R_{3-ma} \mathrel{\widehat{=}} suitable\ data\ representation$, $R_{4-ma} \mathrel{\widehat{=}} visual\ feedback$ are fulfilled. Due to a lack of production data, the performance of our solution with regard to the requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$ for other manufacturability criteria has not yet been proven. However, due to the data-driven structure, this requirement is in principle fulfilled. Through our experiments, we have also shown that our solution is able to learn a geometric manufacturability criterion with only a few hundred 3D models labeled with respect to the corresponding manufacturability criterion. This supports our assumption that with a manageable amount of production data further manufacturability criteria can be learned.

In addition to the general functionality of the *Manufacturability Analysis*, we have also shown that our solution is able to evaluate 3D models and generate the visual feedback on average in about 100 seconds. Therefore, also the requirement $R_{5-ma} \mathrel{\widehat{=}} computation\ time$ is fulfilled, what allows our solution for the *Manufacturability Analysis* of 3D models to be integrated into the web platform-based ordering process of AM service providers without unnecessary delays. Therefore, our solution offers the possibility to detect non-manufacturable components directly in the ordering process. Thus, the customers have the option to revise their 3D model directly without delaying the ordering process by waiting unnecessarily in the order queue as described in Section 2.2. With the help of our solution, it is therefore possible to support the ordering process of customers and thus reduce the average total duration of the process chain of AM service providers.

10.1.2 3D Component Recognition

For the process step *3D Component Recognition*, we developed our solution with regard to the requirements $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{3-cr} \mathrel{\widehat{=}} adaptability$, $R_{4-cr} \mathrel{\widehat{=}} process\ time$, $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$. The first three requirements have to be met to provide the general functionality of the *3D Component Recognition*, the

last three requirements are relevant to enable the solution to be integrable into the process chain of AM service providers and to enable an economical operation of the solution.

Our solution consists of the two main building block of the multi-view-image-based CNN architecture RotationNet and the generation of physically sound training data. The solution has been designed to meet the requirements $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$, $R_{2-cr} \mathrel{\widehat{=}} robustness$, $R_{3-cr} \mathrel{\widehat{=}} adaptability$ with regard to the general recognition functionality on the one hand and to be directly integrable into the process chain of AM service provider on the other hand. To achieve that, the approach is able to automatically extract all necessary information for the generation of synthetic training data from the digital data process chain of AM service providers, in order to subsequently recognize the produced components using photos of them.

In Chapter 9, we have shown that our solution based on the state-of-the-art MVCNN architecture RotationNet and physically sound training data is able to recognize additively manufactured components with a high accuracy. Despite the absence of physical training images of the components to be recognized, the solution is able to learn to recognize the components using physically sound synthetic training data and transfer learning. For a standard build job which was designed in relation to the build jobs occurring in the real production of AM service providers, our solution is able to recognize the corresponding components with an accuracy of 98.51%. Although the recognition rate decreases with increasing difficulty of the data sets, we have developed an adaptation of our approach that should allow to achieve a consistently high recognition rate independent from the complexity of the build jobs. We will deal with this in the following Section 10.2. By using the data-driven augmentation technique of physically sound training data and a combination of photometric augmentation techniques, our solution is able to achieve these high recognition rates independent of changes in the image generation. Thus, our solution has shown that it meets the requirements $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$ and $R_{2-cr} \mathrel{\widehat{=}} robustness$.

The data-driven approach also enables our solution to adapt on a daily basis to the varying components being produced. Based on the automatically generated training data, for each build job a separate instance of our solution is fine-tuned based on a pre-trained RotationNet instance, where we can always guarantee the optimal performance of our solution. Thus, the requirement $R_{3-cr} \mathrel{\widehat{=}} adaptability$ is fulfilled by our solution to the full extent.

Currently, our solution is able to adapt to standard build jobs as they occur in the daily production within the defined time frame. With the changes described in section 9.3, this is also possible for more complex build jobs and thus the requirement $R_{4-cr} \mathrel{\widehat{=}} process\ time$ can be fulfilled. Due to the comparatively cheap hardware necessary for an image-based system, the also the requirements $R_{5-cr} \mathrel{\widehat{=}} scalability$ and $R_{6-cr} \mathrel{\widehat{=}} process\ costs$ are fulfilled by our solution.

Another important aspect is the integration of our solution into the process chain of AM service providers. To ensure that our solution can be used at AM service providers, the daily training processes of the solution instances must be executed fully automated, without the necessity of any manual steps. Our solution is structured in such a way that all necessary information is extracted automatically from the digital data process chain of AM service providers. Therefore, no adjustments to the AM service provider's processes are necessary. For each build job, an instance of our solution can be trained in a fully automated way, which is subsequently able to recognize the corresponding components as soon as they have been produced. Our solution can therefore be used to replace the currently very high manual effort with an automated solution. This enables AM service providers to save both time and costs.

Within this thesis, we have shown that it is possible to automate the complex and especially variable process-steps of *Manufacturability Analysis* for AM and *3D Component Recognition* using adaptive solutions. Due to its continuous linking of the digital data process chain and the physical manufacturing process chain, the process chain of AM service providers offers the opportunity to use data-driven solutions to automate sub-processes that can not or only to a limited extent be automated with traditional methods. As a result, data-driven solutions, like the ones we developed, can enable AM service providers to leverage their strengths compared to traditional manufacturing techniques even with steadily growing production numbers by optimizing the process chain.

In the following final section of this thesis, we will discuss further steps that we believe are necessary for proceeding with our previous work.

10.2 Future Work

The experiments, which we have carried out, have shown that our solutions have the potential for being used in the process chain of AM service providers. Nevertheless, the experiments have also shown some weaknesses of our solutions which must be worked on.

10.2.1 Manufacturability Analysis

In Chapter 8, we already described that the current state of our HCNN-based solution is not invariant to rotations of the 3D models. Therefore, processing the same 3D model in different orientations, can lead to different classification results. Currently a critical geometry is not necessarily detectable in all orientations. To overcome this problem, a solution which is more invariant to rotations is necessary. As already explained in Section 8.3, a joint "multi-view" processing of the hst-files created in different orientations would likely lead to a more robust

classification. This would allow us to further strengthen the performance of our solution with respect to the requirement $R_{2-ma} \mathrel{\widehat{=}} accuracy$.

Additionally, for the *Manufacturability Analysis*, further manufacturability criteria have to be learned by our solution. To the current state, we have trained the solution based on a synthetically generated data set which is labeled with regard to the *minimum wall thickness* criterion. With our experiments, we have shown that our solution is able to learn a geometric criterion relevant for the manufacturability, using an appropriately labeled data set. However, our goal is to learn additional manufacturability criteria based on real production data. Therefore, once sufficient data for another criterion has been collected by the AM service provider, which we used as reference, our approach will be trained based on this data. Thereby, we want to show that our solution also meets the requirement $R_{1-ma} \mathrel{\widehat{=}} adaptability$.

10.2.2 3D Component Recognition

Further steps are also necessary for the *3D Component Recognition*. To the current state, all experiments have been carried out using a hand-held smart phone camera. As stated in Section 9.2, that leads to variances in the creation of the physical inference data. Thus, they are not optimally aligned to the synthetic training data. Constructing the already designed recognition station would enable an optimized inference data generation, potentially further optimizing the performance of our solution with regard to the requirement $R_{1-cr} \mathrel{\widehat{=}} recognition\ rate$. Additionally, in Section 9.3, we have proposed a method to divide the data sets of the build jobs in sub-sets what enables us to solve multiple easier recognition problems instead of the trying to recognize the whole data set as a single entity. By conducting further experiments regarding the splitting of the data sets, we could verify whether it is consistently possible to achieve a recognition rate significantly closer to 100 percent.

With our work on the two process steps of *Manufacturability Analysis* for AM and *3D Component Recognition* for additively manufactured components, we have shown that it is possible to automate parts of the complex process chain of AM service providers with the help of data-driven solutions. However, as part of our analysis of the process chain in Chapter 2, we have also identified other process steps that offer the potential for automation with similar solution approaches.

As an example, the process step of *Production Cost Calculation* is very similar to the *Manufacturability Analysis*, since it is difficult to establish concrete mathematical definitions based on which the costs can be determined. The time and thus the costs for work steps such as support removal subsequent to the SLM process, can vary greatly for different geometries.

With the help of our HCNN-based solution, the costs for this or other work steps can possibly be predicted more accurately on the basis of existing process data than is possible on the basis of current calculations. Therefore, we see a lot of potential for the use of data-driven solutions for a further optimization of the process chain of AM service providers in the future.

References

[1] Adam, G. A. and Zimmer, D. (2014). Design for Additive Manufacturing—Element Transitions and Aggregated Structures. *CIRP Journal of Manufacturing Science and Technology*, 7(1):20–28.

[2] Adam, G. A. and Zimmer, D. (2015). On Design for Additive Manufacturing: Evaluating Geometrical Limitations. *Rapid Prototyping Journal*.

[3] Adam, G. A., Zimmer, D., Müller, M., Systems, L. M. N., and Al., E. (2014). Extension of prior developed design rules' range of validity for different boundary conditions in laser sintering. In *ASPE 2014 Spring Topical Meeting: Dimensional Accuracy and Surface Finish in Additive Manufacturing*, pages 30–35. American Society for Precision Engineering, ASPE Berkeley, CA.

[4] Ali, J., Khan, R., Ahmad, N., and Maqsood, I. (2012). Random Forests and Decision Trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272.

[5] Alpaydin, E. (2019). *Maschinelles Lernen*. Walter de Gruyter GmbH & Co KG.

[6] amvision (2018). AM-Vision: 3D-part Recognition. Available at https://am-flow.com/vision/, Last accessed on 2020-12-17.

[7] amvision2 (2018). AM-Vision: 3D-part Recognition. Available at https://am-flow.com/am-flow-and-oceanz-aim-to-automate-am/, Last accessed on 2020-12-17.

[8] Analytics, M. (2021). Manufacturing Analytics Anwenderkonferenz. Available at https://events.gito.de/manufacturing-analytics2019, last accessed on 2021-01-22.

[9] Ansys (2020). Anysy Additive Manufacturing Simulation. Available at https://www.ansys.com/products/structures/additive-manufacturing, last accessed on 2020-09-15.

[10] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward responsible AI. *Information Fusion*, 58:82–115.

[11] Attaran, M. (2017). The Rise of 3-D Printing: The Advantages of Additive Manufacturing over Traditional Manufacturing. *Business Horizons*, 60(5):677–688.

[12] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On Pixel-wise Explanations for Non-linear Classifier Decisions by Layer-wise Relevance Propagation. *PloS one*, 10(7):e0130140.

[13] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

[14] Balu, A., Ghadai, S., Lore, K. G., Young, G., Krishnamurthy, A., and Sarkar, S. (2016). Learning Localized Geometric Features using 3D-CNN: An Application to Manufacturability Analysis of Drilled Holes. *arXiv preprint arXiv:1612.02141*.

[15] Balu, A., Ghadai, S., Young, G., Sarkar, S., and Krishnamurthy, A. (2017). A Machine-learning Framework for Design for Manufacturability. *arXiv preprint arXiv:1703.01499*.

[16] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded Up Robust Features. In *European conference on computer vision*, pages 404–417. Springer.

[17] Biau, G. (2012). Analysis of a Random Forests Model. *The Journal of Machine Learning Research*, 13(1):1063–1095.

[18] Binns, R. (2018). Fairness in Machine Learning: Lessons from Political Philosophy. In *Conference on Fairness, Accountability and Transparency*, pages 149–159.

[19] Blender (2020). Blender Graphics Software. Available at https://www.blender.org/, last accessed on 2020-11-29.

[20] Böhle, M., Eitel, F., Weygandt, M., and Ritter, K. (2019). Layer-wise Relevance Propagation for Explaining Deep Neural Network Decisions in MRI-based Alzheimer's Disease Classification. *Frontiers in aging neuroscience*, 11:194.

[21] Bronstein, M. M. and Kokkinos, I. (2010). Scale-invariant Heat Kernel Signatures for non-rigid Shape Recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1704–1711. IEEE.

[22] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., and Kalinin, A. A. (2020). Albumentations: Fast and Flexible Image Augmentations. *Information*, 11(2):125.

[23] Campbell, I., Diegel, O., Kowen, J., and Wohlers, T. (2018). *Wohlers Report 2018: 3D Printing and Additive Manufacturing State of the Industry: Annual Worldwide Progress Report*. Wohlers Associates.

[24] Chen, J., Fang, Y., and Cho, Y. K. (2018). Performance Evaluation of 3D Descriptors for Object Recognition in Construction Applications. *Automation in Construction*, 86:44–52.

[25] Ciobota, N.-d. (2012). Standard Tessellation Language in Rapid Prototyping Technology. *National Institute of Research and Development for Mechatronics and Measurement Technique, Bucuresti, The Scientific Bulletin of Valahia University–Materials and Mechanics*, 7.

[26] Dick&Dick (2020). Dick & Dick Generative Fertigung. Available at https://www.dick-dick.de/generative-fertigung.html, last accessed on 2020-10-29.

[27] DMRC (2021). Direct Manufacturing Research Center. Available at https://dmrc.uni-paderborn.de/, last accessed on 2021-01-22.

[28] Eitel, F., Soehler, E., Bellmann-Strobl, J., Brandt, A. U., Ruprecht, K., Giess, R. M., Kuchling, J., Asseyer, S., Weygandt, M., Haynes, J.-D., et al. (2019). Uncovering Convolutional Neural Network Decisions for Diagnosing Multiple Sclerosis on Conventional MRI using Layer-wise Relevance Propagation. *NeuroImage: Clinical*, 24:102003.

[29] Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot Learning of Object Categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

[30] Ghadai, S., Balu, A., Sarkar, S., and Krishnamurthy, A. (2018). Learning Localized Features in 3D CAD Models for Manufacturability Analysis of Drilled Holes. *Computer Aided Geometric Design*, 62:263–275.

[31] Ghahramani, Z. (2003). Unsupervised Learning. In *Summer School on Machine Learning*, pages 72–112. Springer.

[32] Gibson, I., Goenka, G., Narasimhan, R., and Bhat, N. (2010). Design Rules for Additive Manufacture. In *Solid Freeform Fabrication Symposium*, pages 705–716. University Of Texas Austin, TX.

[33] Gibson, I., Rosen, D. W., Stucker, B., et al. (2014). *Additive Manufacturing Technologies*, volume 17. Springer.

[34] Glassner, A. S. (1989). *An Introduction to Ray Tracing*. Elsevier.

[35] Glorot, X. and Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[36] GmbH, P. C. (2021). Phoenix Contact GmbH. Available at https://www.phoenixcontact.com/online/portal/de?1dmy&urile=wcm%3apath%3a/dede/web/home, last accessed on 2021-01-22.

[37] Goguelin, S., Colaco, J., Dhokia, V., and Schaefer, D. (2017). Smart Manufacturability Analysis for Digital Product Development. *Procedia CIRP*, 60:56–61.

[38] Gokuldoss, P. K., Kolla, S., and Eckert, J. (2017). Additive Manufacturing Processes: Selective Laser Melting, Electron Beam Melting and Binder Jetting—Selection Guidelines. *Materials*, 10(6):672.

[39] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning*, volume 1. MIT press Cambridge.

[40] Graves, A. (2012). Supervised Sequence Labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer.

[41] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

[42] Grezmak, J., Zhang, J., Wang, P., Loparo, K. A., and Gao, R. X. (2019). Interpretable Convolutional Neural Network Through Layer-wise Relevance Propagation for Machine Fault Diagnosis. *IEEE Sensors Journal*, 20(6):3172–3181.

[43] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A Survey of Methods for Explaining Black Box Models. *ACM computing surveys (CSUR)*, 51(5):1–42.

[44] Guillemot, M., Heusele, C., Korichi, R., Schnebert, S., and Chen, L. (2020). Breaking Batch Normalization for better Explainability of Deep Neural Networks through Layer-wise Relevance Propagation. *arXiv preprint arXiv:2002.11018*.

[45] Gunning, D. (2017). Explainable Artificial Intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2:2.

[46] Guo, N. and Leu, M. C. (2013). Additive Manufacturing: Technology, Applications and Research Needs. *Frontiers of Mechanical Engineering*, 8(3):215–243.

[47] Harris, C. G., Stephens, M., et al. (1988). A Combined Corner and Edge Detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer.

[48] Hodaň, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., Urbina, P., Sinha, S. N., and Guenter, B. (2019). Photorealistic Image Synthesis for Object Instance Detection. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE.

[49] Horn, T. J. and Harrysson, O. L. (2012). Overview of Current Additive Manufacturing Technologies and Selected Applications. *Science progress*, 95(3):255–282.

[50] Huang, J.-T., Li, J., Yu, D., Deng, L., and Gong, Y. (2013). Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7304–7308. IEEE.

[51] Industrieanzeiger (2021). Industrieanzeiger. Available at https://industrieanzeiger.industrie.de/technik/machine-learning-beschleunigt-3d-druck/, last accessed on 2021-01-22.

[52] Inoue, T., Choudhury, S., De Magistris, G., and Dasgupta, S. (2018). Transfer Learning from Synthetic to Real Images using Variational Autoencoders for Precise Position Detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2725–2729. IEEE.

[53] Inspect (2021). Inspect: World of Vision. Available at https://www.inspect-online.com/topstories/automation/bildverarbeitung-und-ki-sortieren-3d-gedruckte-bauteile-automatisch, last accessed on 2021-01-22.

[54] Ioannidou, A., Chatzilari, E., Nikolopoulos, S., and Kompatsiaris, I. (2017). Deep Learning Advances in Computer Vision with 3D Data: A Survey. *ACM Computing Surveys (CSUR)*, 50(2):1–38.

[55] Iwana, B. K., Kuroki, R., and Uchida, S. (2019). Explaining Convolutional Neural Networks using Softmax Gradient Layer-wise Relevance Propagation. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4176–4185. IEEE.

[56] Kanezaki, A., Matsushita, Y., and Nishida, Y. (2018). Rotationnet: Joint Object Categorization and Pose Estimation using Multiviews from Unsupervised Viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019.

[57] Khajavi, S. H., Partanen, J., and Holmström, J. (2014). Additive Manufacturing in the Spare Parts Supply Chain. *Computers in industry*, 65(1):50–63.

[58] Kim, G. B., Lee, S., Kim, H., Yang, D. H., Kim, Y.-H., Kyung, Y. S., Kim, C.-S., Choi, S. H., Kim, B. J., Ha, H., et al. (2016). Three-Dimensional Printing: Basic Principles and Applications in Medicine and Radiology. *Korean journal of radiology*, 17(2):182–197.

[59] Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., and Klein, M. (2002). *Logistic Regression*. Springer.

[60] Kohlbrenner, M., Bauer, A., Nakajima, S., Binder, A., Samek, W., and Lapuschkin, S. (2020). Towards Best Practice in Explaining Neural Network Decisions with LRP. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

[61] Konstruktionspraxis (2021). Konstruktionspraxis. Available at https://www.konstruktionspraxis.vogel.de/, last accessed on 2021-01-22.

[62] Kranz, J., Herzog, D., and Emmelmann, C. (2015). Design Guidelines for Laser Additive Manufacturing of Lightweight Structures in TiAl6V4. *Journal of Laser Applications*, 27(S1):S14001.

[63] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6):84–90.

[64] Kruth, J.-P., Mercelis, P., Vaerenbergh, J. V., Froyen, L., and Rombouts, M. (2005). Binding Mechanisms in Selective Laser Sintering and Selective Laser Melting. *Rapid prototyping journal*, 11(1):26–36.

[65] Kumar, S., Choudhary, A., Singh, A. K., and Gupta, A. K. (2016). A Comparison of Additive Manufacturing Technologies. *IJIRST-International Journal for Innovative Research in Science & Technology*, 3(01):06.

[66] Langlotz, C. P., Allen, B., Erickson, B. J., Kalpathy-Cramer, J., Bigelow, K., Cook, T. S., Flanders, A. E., Lungren, M. P., Mendelson, D. S., Rudie, J. D., et al. (2019). A Roadmap for Foundational Research on Artificial Intelligence in Medical Imaging: from the 2018 NIH/RSNA/ACR/The Academy Workshop. *Radiology*, 291(3):781–791.

[67] Lapuschkin, S. (2019). Opening the Machine Learning Black Box with Layer-wise Relevance Propagation.

[68] Lapuschkin, S., Binder, A., Montavon, G., Muller, K.-R., and Samek, W. (2016a). Analyzing Classifiers: Fisher Vectors and Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2912–2920.

[69] Lapuschkin, S., Binder, A., Montavon, G., Müller, K.-R., and Samek, W. (2016b). The LRP Toolbox for Artificial Neural Networks. *The Journal of Machine Learning Research*, 17(1):3938–3942.

[70] Le Bourhis, F., Kerbrat, O., Hascoet, J.-Y., and Mognol, P. (2013). Sustainable Manufacturing: Evaluation and Modeling of Environmental Impacts in Additive Manufacturing. *The International Journal of Advanced Manufacturing Technology*, 69(9-12):1927–1939.

[71] LeCun, Y., Bengio, Y., et al. (1995). Convolutional Networks for Images, Speech, and Time Series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[72] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *nature*, 521(7553):436–444.

[73] Lee, T. (2017). Blender Phong: Multi-view Phong Shading. Available at https://github.com/WeiTang114/BlenderPhong, last accessed on 2020-11-29.

[74] Lefebvre, S. and Hoppe, H. (2006). Perfect Spatial Hashing. *ACM Transactions on Graphics (TOG)*, 25(3):579–588.

[75] Li, H., Tian, Y., Mueller, K., and Chen, X. (2019). Beyond Saliency: Understanding Convolutional Neural Networks from Saliency Prediction on Layer-wise Relevance Propagation. *Image and Vision Computing*, 83:70–86.

[76] Liu, Y., Fan, B., Xiang, S., and Pan, C. (2019). Relation-Shape Convolutional Neural Network for Point Cloud Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904.

[77] Loncomilla, P., Ruiz-del Solar, J., and Martínez, L. (2016). Object Recognition using Local Invariant Features for Robotic Applications: A Survey. *Pattern Recognition*, 60:499–514.

[78] Lowe, D. G. (1999). Object Recognition from Local Scale-invariant eatures. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee.

[79] Luo, R. C. and Kuo, C.-W. (2016). Intelligent seven-DoF Robot with Dynamic Obstacle Avoidance and 3-D Object Recognition for Industrial Cyber–physical Systems in Manufacturing Automation. *Proceedings of the IEEE*, 104(5):1102–1113.

[80] Mani, M., Witherell, P., and Jee, H. (2017). Design Rules for Additive Manufacturing: A Categorization. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 58110, page V001T02A035. American Society of Mechanical Engineers.

[81] Marcelino, P. (2020). Transfer learning from pre-trained models. Available at https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751, last accessed on 2020-12-07.

[82] Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., and Malossi, C. (2018). Bagan: Data Augmentation with Balancing GAN. *arXiv preprint arXiv:1803.09655*.

[83] Materialise (2020). Materialise Robot. Available at https://www.materialise.com/de/software/robot, last accessed on 2020-10-29.

[84] Melchels, F. P., Feijen, J., and Grijpma, D. W. (2010). A Review on Stereolithography and its Applications in Biomedical Engineering. *Biomaterials*, 31(24):6121–6130.

[85] Mikolajczyk, K. and Schmid, C. (2001). Indexing based on Scale Invariant Interest Points. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 525–531. IEEE.

[86] Mitchell, T. (1997). Introduction to Machine Learning. *Machine Learning*, 7:2–5.

[87] Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. (2019). Layer-wise Relevance Propagation: An Overview. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209. Springer.

[88] Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. (2017). Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition. *Pattern Recognition*, 65:211–222.

[89] Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for Interpreting and Understanding Deep Deural Networks. *Digital Signal Processing*, 73:1–15.

[90] Mukherjee, T., Zhang, W., and DebRoy, T. (2017). An improved Prediction of Residual Stresses and Distortion in Additive Manufacturing. *Computational Materials Science*, 126:360–372.

[91] Munoz, A. (2014). Machine Learning and Optimization. *URL: https://www.cims.nyu.edu/˜munoz/files/ml_optimization.pdf [accessed 2020-11-15][WebCite Cache ID 6fiLfZvnG]*.

[92] Murr, L. (2018). Additive Manufacturing of Biomedical Devices: An Overview. *Materials technology*, 33(1):57–70.

[93] Nelaturi, S., Kim, W., and Kurtoglu, T. (2015). Manufacturability Feedback and Model Correction for Additive Manufacturing. *Journal of Manufacturing Science and Engineering*, 137(2).

[94] Nickchen, T., Engels, G., and Lohn, J. (2020). Opportunities of 3D Machine Learning for Manufacturability Analysis and Component Recognition in the Additive Manufacturing Process Chain. In *International Conference on Additive Manufacturing in Products and Applications*, pages 37–51. Springer.

[95] Nickchen, T., Heindorf, S., and Engels, G. (2021). Generating Physically Sound Training Data for Image Recognition of Additively Manufactured Parts. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1994–2002.

[96] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724.

[97] Pan, S. J. and Yang, Q. (2009). A Survey on Transfer Learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[98] Park, S.-H., Yang, D.-Y., and Lee, K.-S. (2009). Two-Photon Stereolithography for Realizing Ultraprecise Three-dimensional Nano/Microdevices. *Laser & Photonics Reviews*, 3(1-2):1–11.

[99] Patel, M. N. and Tandel, P. (2016). A Survey on Feature Extraction Techniques for Shape Based Object Recognition. *International Journal of Computer Applications*, 137(6):16–20.

[100] Pereira, T., Kennedy, J. V., and Potgieter, J. (2019). A Comparison of Traditional Manufacturing vs Additive Manufacturing, the best Method for the Job. *Procedia Manufacturing*, 30:11–18.

[101] Protiq (2020). Protiq GmbH. Available at https://www.protiq.com/, last accessed on 2020-09-08.

[102] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

[103] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. (2016). Volumetric and Multi-view CNNs for Object Classification on 3D Data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

[104] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in neural information processing systems*, pages 5099–5108.

[105] Riegler, G., Osman Ulusoy, A., and Geiger, A. (2017). Octnet: Learning Deep 3D Representations at High Resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586.

[106] Rokach, L. and Maimon, O. (2005). Decision Trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer.

[107] Rosten, E., Porter, R., and Drummond, T. (2008). Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–119.

[108] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient Alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. Ieee.

[109] Rudolph, J.-P. and Emmelmann, C. (2017a). Analysis of Design Guidelines for Automated Order Acceptance in Additive Manufacturing. *Procedia CIRP*, 60:187–192.

[110] Rudolph, J.-P. and Emmelmann, C. (2017b). A Cloud-based Platform for Automated Order Processing in Additive Manufacturing. *Procedia Cirp*, 63:412–417.

[111] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3):211–252.

[112] Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. (2016). Evaluating the Visualization of what a Deep Neural Network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

[113] Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. (2020). Toward Interpretable Machine Learning: Transparent Deep Neural Networks and Beyond. *arXiv preprint arXiv:2003.07631*.

[114] Sánchez, J. and Perronnin, F. (2011). High-dimensional Signature Compression for Large-scale Image Classification. In *CVPR 2011*, pages 1665–1672. IEEE.

[115] Sankaranarayanan, S., Balaji, Y., Jain, A., Nam Lim, S., and Chellappa, R. (2018). Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3752–3761.

[116] Schuster, M. (1999). On Supervised Learning from Sequential Data with Applications for Speech Recognition. *Nara Institute of Science and Technology*.

[117] Seifert, C., Aamir, A., Balagopalan, A., Jain, D., Sharma, A., Grottel, S., and Gumhold, S. (2017). Visualizations of Deep Neural Networks in Computer Vision: A survey. In *Transparent Data Mining for Big and Small Data*, pages 123–144. Springer.

[118] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

[119] Shao, T., Yang, Y., Weng, Y., Hou, Q., and Zhou, K. (2018). H-CNN: Spatial Hashing Based CNN for 3D Shape Analysis. *IEEE transactions on visualization and computer graphics*.

[120] Shapeways (2020). Shapeways 3D Printing Service. Available at https://www.shapeways.com/, last accessed on 2020-10-29.

[121] Sharma, H. (2020). Transfer learning : Approaches and empirical observations. Available at https://hackernoon.com/transfer-learning-approaches-and-empirical-observations-efeff9dfeca6, last accessed on 2020-12-07.

[122] Shi, Y., Zhang, Y., Baek, S., De Backer, W., and Harik, R. (2018). Manufacturability Analysis for Additive Manufacturing using a Novel Feature Recognition Technique. *Computer-Aided Design and Applications*, 15(6):941–952.

[123] Shorten, C. and Khoshgoftaar, T. M. (2019). A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60.

[124] Siemens (2020). Siemens Additive Manufacturing Simulation. Available at https://www.plm.automation.siemens.com/global/de/webinar/additive-manufacturing/61251, last accessed on 2020-09-15.

[125] Simufact (2020). Simufact Additive. Available at https://www.simufact.de/simufact-additive.html, last accessed on 2020-09-15.

[126] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-View Convolutional Neural Networks for 3D Shape Recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953.

[127] Su, J.-C., Gadelha, M., Wang, R., and Maji, S. (2018). A Deeper Look at 3D Shape Classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0.

[128] Sun, J., Ovsjanikov, M., and Guibas, L. (2009). A Concise and Brovably Informative multi-scale Signature based on Heat Diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library.

[129] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press.

[130] Tagliaferri, V., Trovalusci, F., Guarino, S., and Venettacci, S. (2019). Environmental and Economic Analysis of FDM, SLS and MJF Additive Manufacturing Technologies. *Materials*, 12(24):4161.

[131] Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017). Octree Generating Networks: Efficient Convolutional Architectures for High-Resolution 3D Outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096.

[132] Tech, R. (2021). Rapid Tech Conference. Available at https://www.rapidtech-3d.de/, last accessed on 2021-02-04.

[133] Tedia, S. and Williams, C. B. (2016). Manufacturability Analysis Tool for Additive Manufacturing using Voxel-based Geometric Modeling. In *27th annual international solid freeform fabrication (SFF) symposium*, pages 3–22.

[134] Telea, A. and Jalba, A. (2011). Voxel-based Assessment of Printability of 3D Shapes. In *International symposium on mathematical morphology and its applications to signal and image processing*, pages 393–404. Springer.

[135] Thrun, S. and Pratt, L. (2012). *Learning to Learn*. Springer Science & Business Media.

[136] Tjoa, E. and Guan, C. (2019). A survey on Explainable Artificial Intelligence (XAI): towards medical XAI. *arXiv preprint arXiv:1907.07374*.

[137] Toshev, A., Makadia, A., and Daniilidis, K. (2009). Shape-based Object Recognition in Videos using 3D Synthetic Object Models. In *2009 IEEE conference on computer vision and pattern recognition*, pages 288–295. IEEE.

[138] Toshev, A., Taskar, B., and Daniilidis, K. (2012). Shape-based Object Detection via Boundary Structure Segmentation. *International journal of computer vision*, 99(2):123–146.

[139] Uçar, A., Demir, Y., and Güzeliş, C. (2016). Moving Towards in Object Recognition with Deep Learning for Autonomous Driving Applications. In *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–5. IEEE.

[140] Ulu, E., Gecer Ulu, N., Hsiao, W., and Nelaturi, S. (2020). Manufacturability Oriented Model Correction and Build Direction Optimization for Additive Manufacturing. *Journal of Mechanical Design*, 142(6).

[141] VDMA (2021). Verein Deutscher Maschinen- und Anlagenbauer. Available at https://www.vdma.org/en/, last accessed on 2021-01-22.

[142] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X. (2017a). O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics (TOG)*, 36(4):1–11.

[143] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X. (2017b). O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics (SIGGRAPH)*, 36(4).

[144] Wimpenny, D. I., Pandey, P. M., and Kumar, L. J. (2017). *Advances in 3D Printing & Additive Manufacturing Technologies*. Springer.

[145] Wohlers, T. T., Campbell, I., Diegel, O., Huff, R., and Kowen, J. (2020). *Wohlers Report 2020*. Wohlers Associates, Inc.

[146] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D Shapenets: A Deep Representation for Volumetric Shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

[147] Xie, N., Ras, G., van Gerven, M., and Doran, D. (2020). Explainable Deep Learning: A Field Guide for the Uninitiated. *arXiv preprint arXiv:2004.14545*.

[148] Yang, Y., Tresp, V., Wunderle, M., and Fasching, P. A. (2018). Explaining Therapy Predictions with Layer-wise Relevance Propagation in Neural Networks. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 152–162. IEEE.

[149] Zellers, R., Bisk, Y., Farhadi, A., and Choi, Y. (2019). From Recognition to Cognition: Visual Commonsense Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6720–6731.

[150] Zhang, Y., Yang, S., and Zhao, Y. F. (2020). Manufacturability Analysis of Metal Laser-based Powder Bed Fusion Additive Manufacturing—A Survey. *The International Journal of Advanced Manufacturing Technology*, 110(1):57–78.

[151] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

[152] Zhou, Q. and Jacobson, A. (2016). Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797*.

[153] Zhu, X. J. (2005). Semi-supervised Learning Literature Survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

[154] Zienkiewicz, O. C., Taylor, R. L., Taylor, R. L., and Taylor, R. L. (2000). *The Finite Element Method: Solid Mechanics*, volume 2. Butterworth-heinemann.