Near Design Analyse of Structural Components for Additive Manufacturing

Stefan Dahlke*[1], Stefan Holtzhausen[1], Philipp Sembdner[1], Christine Schöne[1], Ralph Stelzer[1], Richard Kordaß[2]

[1] Dresden University of Technology, Germany
[2] Fraunhofer IWU, Dresden, Germany

* Corresponding Author: stefan.dahlke@tu-dresden.de, +49 351 46333580

Abstract

Additive manufacturing (AM) processes still pose challenges for engineers. However, there are a large number of process-related manufacturing restrictions for metallic powder-bed processes. These limitations must be considered by the designer, which requires a high level of process knowledge. With common design programs (CAD) it is not possible to make a decision about the manufacturability of a component. Therefore, a requirements analysis of the needed information was carried out. A total of 15 functions were determined, which are resolved in order to be able to make a qualified statement. This work concerned with the development of an analysis tool, which receives data from CAD software and evaluates the component. A combination of two different discrete data sets (triangle mesh, volume model) is used to analyse the component. Based on these both data sets and the usage of a cost field algorithm, the required information can be calculated to evaluate the component.

1 Introduction

In the field of additive manufacturing, various manufacturing processes are increasingly establishing themselves, which differ with regard to the technology used (e. g. selective laser melting or electron beam melting in the metal sector) or the materials to be processed (e. g. metallic or ceramic materials) [1]. Due to their specific process characteristics, these techniques have different requirements on the geometry of the component to be manufactured, which must be taken into account during the design process [2, 3]. The higher the consideration of the requirements in the design process, the less effort is involved in preparation (pre-process), e.g. for creating support structures. Furthermore, it is assumed that a production-oriented design improves the quality of the components' results and reduces manufacturing costs [4]. Therefore, a preliminary analysis of the component according to the additive manufacturing process to be used is necessary.

However, there are few methods and tools available for a fast and efficient production analysis of a component that is close to the design phase. For example *Martha* has developed an additive manufacturing tool for several CAD programs [5]. Up to now, the components have been inspected for buildability, if at all, very close to production in the pre-process [6].

The aim is therefore to develop methods and tools that help the designer to assess the modelled component in terms of its manufacturability. This auxiliary tool must be completely integrated into the process chain for additive manufacturing respectively the existing design tool (see Figure 1). At the same time, however, neutrality with regard to the CAD system to be used must be maintained. This also supports distributed product development and development via e.g. manufacturing service providers, where different CAD systems can be used. Communication with the CAD system is carried out via a corresponding link. With regard to the manufacturing technology used, the necessary parameters are stored in the analysis tool via defined configurations (templates).

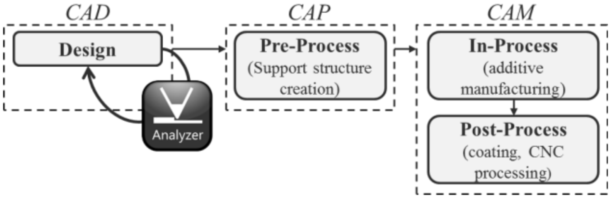


Figure 1: AM-process chain after VDI Guideline 3405 with near design analyse integration [2]

The necessary information must be available to the designer without delay. Therefore, no computationally intensive simulations or optimizations are carried out, but the methods to be developed and implemented concentrate solely on the analysis of the geometry of the component.

2 Needs analysis

In order to determine the required functionalities, a needs analysis was carried out among users from research and industry, whose applications include classical single part constructions as well as special component modelling, on the basis of a defined application case. The selection of the functions to be implemented was made by evaluating or prioritizing the functions mentioned above. The additive methods

Selective Laser Melting (SLM), Electron Beam Melting (EBM), Lithography-based Ceramic Manufacturing (LCM) are examined below. An overview of the needs analysis is shown in Table 1. Marking in brackets means that this does not have to be a mandatory functionality. If this functionality is still available, it can be useful for the process.

Table 1: Functions for component analysis with relevance to the respective manufacturing process

Function	**SLM**	**EBM**	**LCM**
Alignment via building platform	x	x	x
Display multiple construction jobs	x	x	x
Exposure area above mounting height	x	x	
Marking overhangs by angle	x	x	x
Visualize external support structures and collision detection	x	x	
Visualizing the feasibility of support structures (using angles + cantilever width)	x		
Typing overhangs	x	x	x
Local weight	(x)	(x)	
Mark support and component position on board	(x)		
Control / support during support creation			
Minimum wall thicknesses	x	x	x
Smallest drilling vs. minimum drilling	(x)	(x)	(x)
Material accumulations	x		x
Cavities	x	x	x
Build direction analysis (detection of islands)	(x)	(x)	x

3 Hybrid data model

The presented analysis program is based on an underlying hybrid data model. On the one hand, a tesselated surface model (STL model) exported from the design program is used and on the other hand, a discrete volume model. The volume model is a three-dimensional, evenly distributed data structure. An element of the data structure is called voxel. A voxel can have two different stages (air or material). The volume model is generated from the STL model using a slicing algorithm. Based on the alignment, a defined number of layer patterns are generated. Both data models are aligned to each other (Figure 2).

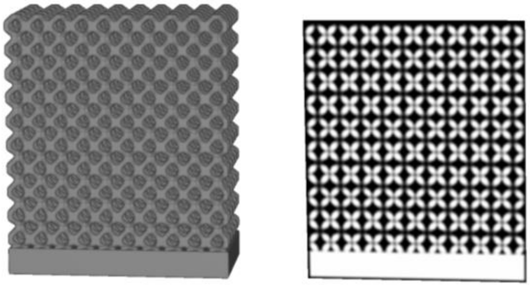


Figure 2: Left: example mesh; Right: slice image from volume dataset

The surface model consisting of tessellated data (triangular mesh) offers the advantage that by using Graphical Processing Units (*GPUs*) geometry-related analyses can be carried out efficiently. The render pipeline (Figure 3) used here represents the functionality of GPUs and can be modified by using shaders at certain points within the pipeline through small programs (shader programs).

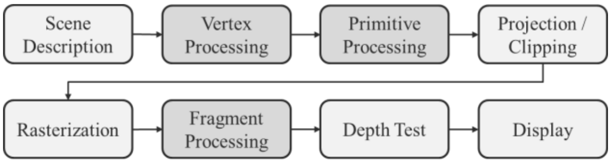


Figure 3: Render pipeline, grey areas are programmable with shaders

Render geometries in the form of points, lines or triangles are used as input parameters. These primitive geometries are projected onto a defined image plane (user's viewing plane) using the projection matrix. Afterwards, the image area is truncated and the resulting geometries are rasterized. As a result of the rasterization, so-called image fragments (later possible image pixels) are present. With the help of a fragment shader it is possible to run a defined program for each fragment. The analysis software compares the respective surface angle with the normal of the building platform within this fragment shader. Depending on the resulting angle, the fragment is coloured using a defined colour scale. The computational effort for the angle calculation in the fragment shader is so low that the determination and representation can be carried out in real time. This enables the designer to position the component freely above the building platform and immediately get the changes of the angle dependencies displayed. However, it should be noted that the respective angles are not stored in any form and must first be read out using other techniques.

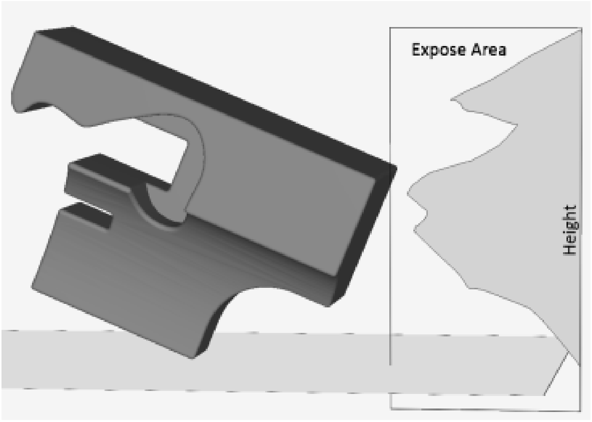


Figure 4: Example visualization of the exposure areas

The volume model offers the possibility to use well-known image processing algorithms for component analysis. On the one hand, it is possible to efficiently determine the exposure areas per cross-section, for example. Based on the occupied voxel within a layer, the cross-sectional area is calculated and can be displayed in a diagram with other exposure areas in the installation direction (Figure 4). On the other hand, a so-called cost-field algorithm can be implemented based on the volume model. Based on this, various analyses can be carried out according to the tasks set out.

3.1 Cost Field Algorithm

A cost field algorithm or Dijkstra algorithm is usually used for routing problems in network technology or robotics [7]. Based on the volume model, a three-dimensional integer array is created. A schematic representation of the algorithm can be found as pseudocode in *Code 1*. At the beginning, each element receives a defined status based on the voxel of the volume model. This is either material (-2) or air area (-1). Afterwards, start nodes (*startNodes*) are determined from one of the two categories. These start nodes can be all air voxels above the defined building platform, for example, to investigate the feasibility of possible support structures. The start nodes receive the value or cost 0 and are defined as nodes to be checked (*checkNodes*).

Code 1: Schematic flow of the cost-field algorithm in pseudocode

```
checkNodes = startNodes
do
  foreach node in checkNodes
    checkNeighbours
    if neighbour -1 || > node+1
      neighbour = node+1
      nextNodes.add(neighbour)
  checkNodes = nextNodes
while checkNodes != null
```

Within an iteration loop, the values of the neighbours are read for each node to be checked (*node*). If the value of a neighbouring node is -1 or greater than the current node value + 1, the value of the neighbouring node is set to the value of the current node + 1. Furthermore, the neighbouring nodes for which a new value has been set are buffered in a list (*nextNodes*). At the end of the iteration loop, this list is defined as nodes to be checked. The iteration loop will continue until there are no more nodes to be checked. This algorithm gives each voxel the cost of the way to the nearest starting point.

Various information can be derived from the resulting cost field. This includes the shortest path and the angle of the individual path segments to the building platform, inner cavities and so-called stalactites. These are overhangs, which do not have a downward connection to the building platform. This information is relevant for the LCM and EBM procedures (see table 1). For these analyses the algorithm has to be changed a little bit. For internal analysis of the component, for example, material voxel is defined as the start node and the neighbouring nodes are checked for the value -2.

3.2 Resampling on the GPU

For the analysis of the component based on the volume model, it must be aligned with the building platform. This means that the X-Y plane of the solid model is parallel to the building platform and the Z coordinate represents the layer plane. The building platform would thus theoretically be located at the Z-position -1, which is practically outside of the data model. As soon as the component is rotated and the alignment to the building platform changes, the solid model must be realigned. Since a new execution of the slicing algorithm would slow down the workflow, it was necessary to find a way to quickly realign the volume model. For this purpose, the techniques of modern GPUs are also used. The solid model is loaded into the GPU as a three-dimensional texture. The 3D texture is re-aligned with a rotating matrix according to the positioning of the component above the building platform. A further 3D texture is defined as the render target, which is aligned parallel to the building platform. This texture has the same voxel resolution as the original 3D texture. The size of the texture is determined based on the axis-aligned bounding box. Within an iteration loop, the layer images of the new 3D texture are described based on the values of the old ones. The loop runs until the complete volume model has been resampled. The resulting 3D texture is read from the graphics card and exchanged with the previous volume model. By using the GPU, this process can be carried out within a few milliseconds depending on the model size.

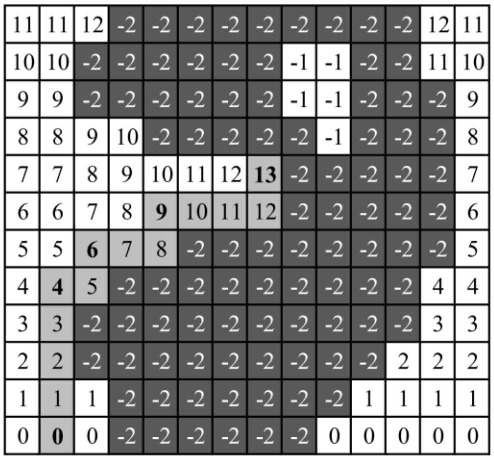
4 Analysis functions

Based on the different cost fields, the component can be analysed for different circumstances. Among other things, this includes the determination of possible paths for the support structure, the weight over nodes, wall thicknesses, overhangs etc. Two example analyses are described in detail at this point.

4.1 Support feasibility

For the analysis of support feasibility, the cost-field algorithm described above is used. The result can be seen schematically in Figure 5.

All voxels of the part are defined with -2 (grey areas), all areas available for support are defined with -1 (white areas). If a distance between the support and part geometry is required, this can be defined beforehand. Around the part, this area would then be defined separately. During the cost field algorithm, all nodes above which a component node is located are stored as potential support nodes (*supportNodes*). The highlighted node 13 is an example of this. Support feasibility for these nodes is analysed. First of all, for each of these nodes, the system checks whether the costs are equal to the Z value (shift number). If this is the case, the support node is located directly above the building platform and the support can be carried out without any problems. If the costs are higher, the support node is located above the part geometry. In this case, the shortest way to the building platform is determined (*Code 2*). This is done by checking the costs around the current node and taking the most favourable node as the next one. If costs are the same for several nodes, the first node found is always used. The algorithm always starts with the node below. If the last node in the path is in the same layer as the current one and the node under the current node is most advantageous, the current node is defined as an edge node (nodes 9, 6, 4 in Figure 5). The path finding is carried out until the building platform is reached. At the end, the support node, all edge nodes and the end node above the building platform are saved as a possible path for support. The angle of all path segments is analysed and color-coded in relation to a defined maximum angle for the support.



11	11	12	-2	-2	-2	-2	-2	-2	-2	-2	-2	12	11
10	10	-2	-2	-2	-2	-2	-2	-1	-1	-2	-2	11	10
9	9	-2	-2	-2	-2	-2	-2	-1	-1	-2	-2	-2	9
8	8	9	10	-2	-2	-2	-2	-2	-1	-2	-2	-2	8
7	7	8	9	10	11	12	**13**	-2	-2	-2	-2	-2	7
6	6	7	8	**9**	10	11	12	-2	-2	-2	-2	-2	6
5	5	**6**	7	8	-2	-2	-2	-2	-2	-2	-2	-2	5
4	**4**	5	-2	-2	-2	-2	-2	-2	-2	-2	-2	4	4
3	3	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	3	3
2	2	-2	-2	-2	-2	-2	-2	-2	-2	-2	2	2	2
1	1	1	-2	-2	-2	-2	-2	-2	-2	1	1	1	1
0	**0**	0	-2	-2	-2	-2	-2	-2	0	0	0	0	0

Figure 5: Sample representation of the cost field during analysis for support feasibility

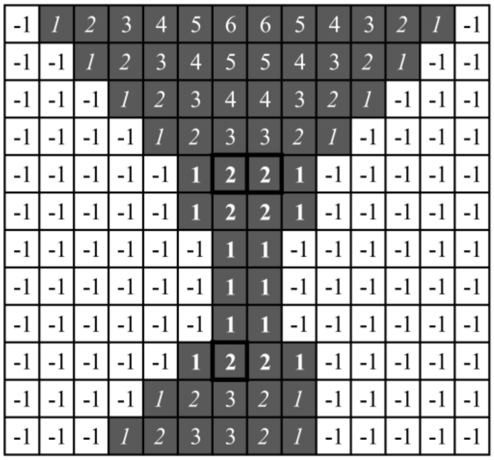
Figure 5 shows that according to the cost-field algorithm, voxel with an initial value (-1) can occur. These voxels represent areas which are cavities within the component. These can be selected easily and the designer can be informed of this issue.

Code 2: Pseudocode representation for determining the shortest path to the building platform

```
supportNodes
edgeNodes
foreach node in supportNodes
  lastZ = node.z
  checkNode = node
  while checkNode.Z != 0
    if checkNode.Cost == checkNode.Z
      path = goStreightDown()
    else
      nextNode = getCheapestNode
      if lastZ = nextNode.Z+1
        edgeNodes.add(checkNode)
      lastZ = checkNode.Z
      checkNode = nextNode
  end
  path = (node, edgeNodes, checkNode)
ende
```

5 Wall thickness analysis

For the three-dimensional analysis of wall thicknesses, the cost-field algorithm is also used. The costs within the component are determined with reference to the component edge voxel. A schematic representation of the result can be seen in Figure 6. All external nodes of the component are defined as start nodes and the corresponding distance costs are calculated using the cost field algorithm. Based on the resulting cost field, thin walls can now be filtered out using a defined minimum wall thickness.



-1	*1*	*2*	3	4	5	6	6	5	4	3	*2*	*1*	-1
-1	-1	*1*	*2*	3	4	5	5	4	3	*2*	*1*	-1	-1
-1	-1	-1	*1*	*2*	3	4	4	3	*2*	*1*	-1	-1	-1
-1	-1	-1	-1	*1*	*2*	3	3	*2*	*1*	-1	-1	-1	-1
-1	-1	-1	-1	-1	**1**	**2**	**2**	**1**	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	**1**	**2**	**2**	**1**	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	**1**	**1**	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	**1**	**1**	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	**1**	**1**	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	**1**	**2**	**2**	**1**	-1	-1	-1	-1	-1
-1	-1	-1	-1	*1*	*2*	3	*2*	*1*	-1	-1	-1	-1	-1
-1	-1	-1	*1*	*2*	3	3	*2*	*1*	-1	-1	-1	-1	-1

Figure 6: Example representation of the cost field for wall thickness analysis

Based on this wall thickness, all nodes within the cost field are selected that have half the wall thickness or less cost (italic numbers). For each of these nodes, the system checks whether there is another node in the neighbourhood of this node that has higher costs. If this is the case, these nodes are located within a sufficiently thick geometry. If the node has no neighbours with higher costs, it is saved in a list. After all nodes have been checked, the list of all saved nodes is processed. For each node within the list, the respective node and the neighbouring nodes up to the component surface are marked. These marked nodes represent all areas where the walls are too thin (bold numbers). This check is direction-dependent. The thick-frame fields in Figure 6 would not be determined if there were no directional analysis. Along the Y-axis, the nodes have a neighbour with higher costs and would therefore drop out of the check, even though the wall thickness along the X-axis is too low.

6 Summary

With the analysis software developed so far, it is currently possible to load any STL model, for example from the CAD system SolidWorks. This is followed by automatic generation of the volume model using the implemented slicing algorithm. Various production-relevant parameters can be set and the respective configuration can be saved and loaded for later use. The component can be positioned freehand above the building platform. The resampling on the graphics card is also implemented and is used to realign the volume model after the positioning of the component. The cost field algorithm and the two analysis functions described above are implemented. Furthermore it is possible to detect cavities, select overhangs (stalactites) and determine the exposure cross section area by the height of the building.

7 Future work

Further functionalities that will be implemented in real time are the aggregation of support nodes to support areas, analysis of holes, material accumulation. Furthermore, it should be possible in the future to load and consider parameters from the CAP process. This includes the display and collision detection of externally generated support structures as well as the positioning of several build jobs. In the future, the designer will be given the opportunity to block areas of the component for support structures or to define preferred ways for support. This is implemented within the data model in such a way that the costs for the voxel are increased within the defined range.

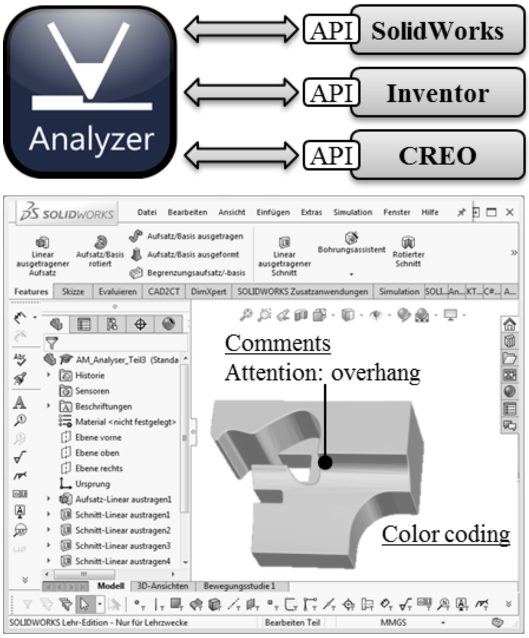


Figure 7: Linking the analyser tool to the CAD system for providing analysis information

The next step in the processing is the continuation of the connection of the neutral analysis tool to the CAD systems used. For this purpose, the accessible programming interface (API) is used. The CAD coupling is initially implemented in the SolidWorks CAD system. Depending on the analysis result, information is provided to the designer directly on the component. These can be, for example, colour coding with regard to possible overhangs, remarks or the representation of the building platform based on the determined installation direction (see Figure 7).

Acknowledgements

This research was supported by the “Zwanzig20 Agent-3D FunGeoS” project (03ZZ0208I). The project is funded by the Federal Ministry of Education and Research (BMBF).

Literature

[1] A. Gebhardt: Additive Fertigungsverfahren: Additive Manufacturing und 3D-Drucken für Prototyping - Tooling – Produktion, München, 2016

[2] VDI Guideline 3405: Additive manufacturing processes, rapid manufacturing: Basics, definitions, processes, Düsseldorf, 2014

[3] M. Süß, B. Klöden, A. Kirchner, T. Weißgärber, D. Hofmann, C. Schöne, R. Stelzer, B. Kieback: Untersuchung zu Konstruktionsempfehlungen für kleine Strukturen beim Elektronenstrahlschmelzen, Proceedings of the 13th Rapid.Tech Conference, Erfurt, 2016, pp. 279-289

[4] S. Danjou, „Mehrzieloptimierung der Bauteilorientierung für Anwendungen der Rapid-Technologie.“ Cuvillier, Göttingen, 2010.

[5] A. M. Martha: Optimierung des Produktentwicklungsprozesses durch CAD-CAM-Integration im Kontext der additiven Fertigung, Dissertation, University of Duisburg-Essen, 2015

[6] Materialise: Materialise Software V21.11 Magics Manual, Leuven, Belgium, 2017

[7] A. Stentz, “Optimal and Efficient Path Planning for Partially-Known Environments”, Carnegie Mellon University; Pittsburgh, 1994