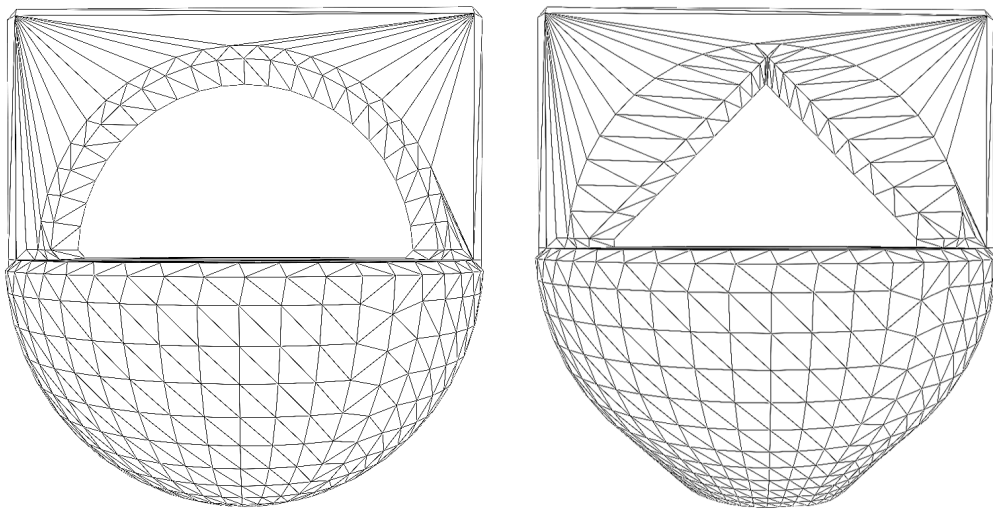




**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

---



# **Automatic geometry alteration when designing for additive manufacturing**

How automated geometry alteration can be utilized to reduce support structures in additive manufacturing

Master's thesis, 2019

Julian Martinsson



# Automatic geometry alteration when designing for additive manufacturing

How automated geometry alteration can be utilized to reduce  
support structures in additive manufacturing

Julian Martinsson



Department of Industrial and Materials Science  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2019

Master's thesis (2019)

© Julian Martinsson 2019.

Supervisor: Olivia Borgue, Department of Industrial and Materials Science  
Examiner: Ola Isaksson, Department of Industrial and Materials Science

Division of Product development  
Chalmers University of Technology  
SE-412 96 Gothenburg  
Telephone +46 31 772 1000

Typeset in L<sup>A</sup>T<sub>E</sub>X  
Gothenburg, Sweden 2019

## **Abstract**

The space industry has developed an interest in converting some of its production from traditional manufacturing to metal additive manufacturing. This may prove beneficial to the space industry due to their generally low production volumes, and their need to minimize product mass. The transition from traditional to additive manufacturing can be difficult, and requires designers to take heed of a new set of design and manufacturing constraints. To aid the designers this thesis project has aimed to create a software which automatically alters the geometry of 3D STL-models. The purpose of the alterations is to reduce the model's need for support structures. Reducing the support structures is necessary to reduce material waste, and to decrease the time required to process the model after it has finished printing. A prototype software was created which can handle simple geometries, but struggles when the model complexity increases. Two main problems were encountered: difficulty to handle overhang surfaces that point directly down, and unintended software behavior which caused issues mostly prevalent in high complexity models. These issues could not be resolved during the course of this project, but could likely be overcome in the future. Finally, this thesis project resulted in the creation of a software prototype capable of altering the geometry of low complexity models.

# Preface

This serves as the punctuation of my time as a student at Chalmers University of technology, where I studied mechanical engineering and product development. Six months of dedication has yielded the master's thesis that lies before you. It has no doubt been rough, but it has also been an interesting experience that I am likely never to forget. And though it sometimes seemed as though I were stumbling through the dark, I in fact always had a guiding light. And for that, I wish to thank Olivia Borgue for her enthusiastic and talented supervision, Ola Isaksson for his commitment and advice, and my family for their support.

# Contents

<b>List of Figures</b>	<b>iv</b>
<b>1 Background</b>	<b>1</b>
1.1 A brief introduction to AM . . . . .	1
1.2 The space industry: a suiting candidate . . . . .	2
1.3 Problems . . . . .	3
1.4 The current toolkit . . . . .	4
1.4.1 Support structure generation . . . . .	4
1.4.2 Orientation optimization . . . . .	4
1.4.3 Automatic geometry alteration . . . . .	4
1.5 Impact on society . . . . .	5
1.5.1 The ethical and societal context . . . . .	5
1.5.2 The potential impact of this project . . . . .	5
<b>2 Aim</b>	<b>7</b>
2.1 Demarcations . . . . .	7
<b>3 Theory</b>	<b>8</b>
3.1 Metal additive manufacturing . . . . .	8
3.1.1 Overhang . . . . .	8
3.1.2 Support structures . . . . .	9
3.2 Computer aided additive manufacturing . . . . .	10
3.2.1 The STL file format . . . . .	10
3.2.2 Voxel representation . . . . .	11
<b>4 Method</b>	<b>12</b>
4.1 Interviews . . . . .	12
4.2 Benchmarks . . . . .	12
4.3 Concept generation and selection . . . . .	13
4.4 Creating the prototype . . . . .	13
<b>5 Results</b>	<b>14</b>
5.1 Interviews . . . . .	14
5.1.1 The problem with orientation optimization . . . . .	14
5.1.2 Strict requirements in space technology . . . . .	14
5.2 Problem detection algorithm . . . . .	14

5.2.1	Generated problem detection concepts . . . . .	15
5.2.2	Evaluation of concepts . . . . .	16
5.2.3	Concept selection and result . . . . .	17
5.3	Basic problem correction algorithm . . . . .	17
5.3.1	Generated correction concepts . . . . .	18
5.3.2	Concept evaluation and selection . . . . .	18
5.3.3	Focus on one face at a time . . . . .	21
5.3.4	Performing the change . . . . .	22
5.3.5	The meta algorithm . . . . .	23
5.3.6	Problems with the basic problem correction . . . . .	25
5.4	Advanced problem correction algorithm . . . . .	26
5.4.1	Addressing the problems with curved geometries . . . . .	26
5.4.2	Dealing with leaks . . . . .	27
5.4.3	Overhang without an angle . . . . .	27
5.5	Final prototype . . . . .	28
5.5.1	User interface . . . . .	29
5.5.2	Performance of the single face algorithm . . . . .	30
5.5.3	Orientation optimization . . . . .	32
5.5.4	Angle injection . . . . .	35
<b>6</b>	<b>Discussion</b>	<b>37</b>
6.1	Geometry alteration and the space industry . . . . .	37
6.2	Further development . . . . .	37
6.2.1	Improving the performance . . . . .	37
6.2.2	Improving the results . . . . .	38
6.2.3	Improving the experience . . . . .	38
6.2.4	Improving integration . . . . .	38
6.3	Use cases . . . . .	39
<b>7</b>	<b>Summary</b>	<b>40</b>
	<b>Bibliography</b>	<b>43</b>
	<b>Appendices</b>	<b>I</b>
<b>A</b>	<b>Timetable</b>	<b>II</b>
<b>B</b>	<b>Benchmarks</b>	<b>III</b>



# List of Figures

1.1	Simplified sketch of a laser powder bed fusion system. . . . .	2
1.2	Area close to the top has darkened due to having close to insufficient heat dispersion. Could have been avoided by reducing the angle (increasing $\varphi$ ), or by introducing support structures. . . . .	3
3.1	Illustration of the overhang angle, and how it relates to the workpiece and the substrate. . . . .	8
3.2	Illustration of three different overhang angles. a: $60^\circ$ , b: $45^\circ$ , c: $25^\circ$ . Overhang a is stable, b is on the edge of stability and instability and c is potentially unstable. . . . .	8
3.3	Support structure of a $0^\circ$ overhang (also referred in this thesis as a "flat overhang"). Notice how the support structure is made of the same material as the part itself. . . . .	9
3.4	The structure of an STL-file. This example contains only two faces, but an STL-file could in theory contain any number of faces. Both binary and ASCII STL-files represents the data using similar architectures (the binary version contains slightly more information which is not relevant to understanding how the file is structured). The colored variant contains slightly more information that is not relevant for this thesis. . . . .	10
3.5	A voxelized cylinder. Each cube represents a voxel. The entire model is a representation of a $30^3$ matrix consisting of ones and zeros. Each element that contains the number "one" is drawn using a cube, while all the other elements that contain zero remain unrendered. . . . .	11
5.1	2-dimensional illustration of a voxelized structure. The original structure is represented by the green (supported faces) and the red (unsupported faces) contour. The greyed out grid elements represents the voxel representation of the original structure. The dark grey voxel lacks proper support, as it does not have an adjacent voxel directly or diagonally underneath it. . . . .	15
5.2	Concept of how machine learning might be able to be used to recognize problematic areas of a model using image recognition. The areas highlighted with the color red would be considered as problematic areas, identified by the machine learning algorithm. . . . .	15

5.3	The first successful problem detection experiment. The Z-direction represents the printing direction. The red faces represent surfaces that possesses an overhang angle that is too small (in this example too small is considered to be $\varphi < 45^\circ$ ). The green surfaces are problem free. This early in development there was no regard for surfaces that touched the ground, which has resulted in the lower "feet" also being registered as problematic surfaces, despite them being supported by the substrate (the substrate is not visible in this figure). . . . .	17
5.4	Voxel conversion information loss. Figure <b>a)</b> is an image of the original STL-file containing the arch-sphere benchmark. Figure <b>b)</b> shows a version of the arch-sphere that has been converted to a voxel format, and then back into STL again. The visibly jagged edges, and the increased face density are results of the imperfect conversion process.	20
5.5	On display is the cylinder benchmark, before and after it passed through the problem correction algorithm. In <b>b)</b> the model is shown after it has passed through the algorithm, which has evened out the bottom section and given the top half of the inside of the cylinder a $45^\circ$ slope. . . . .	21
5.6	This figure depicts the single face algorithm (SFA). Seen in the figure is a single face, and its components: the two roaming vertices, the anchor vertex, and the normal vector. The two red arrows represents the change vectors, which are calculated using the SFA. The change vectors always points in the same direction as the normal vectors projection on the XY-plane, and are later used to resolve problematic overhang angles. In this example the desired minimum overhang is $45^\circ$ . Thus, the change vectors strive to push the roaming vertices in a manner that would result in a $45^\circ$ overhang. . . . .	21
5.7	The meta algorithm. This algorithm shows how the detection algorithm (purple) works together with the correction algorithm (green) in order to produce results. The queue does not keep the faces in any particular order. . . . .	24
5.8	"Blobbing" on a surface with a high face density, seen from two different perspectives. This particular model is of the handle of a teapot.	25
5.9	Rounded surface flattening. If a vertex touches the ground, and the surrounding vertices does not, then the surrounding vertices are brought down to ground level. This could reduce both the need for support structures and the need for machining in post processing. The bright blue dot represents the vertex that is touching the ground, and the green dots represent the surrounding vertices. . . . .	26
5.10	This figure displays the algorithm in use. In this case the algorithm has detected a curved surface where only one vertex touches the ground. It has thus flattened the surrounding vertices, reducing the need for support structures. After flattening the bottom, the SFA pushed out the vertices at the bottom, creating a stable $45^\circ$ overhang.	27
5.11	The final version of the prototype. . . . .	29

5.12	A diagram of how the weight of a face is determined based on the overhang angle. The value is multiplied by the area of the face, thus giving larger overhang surfaces a larger penalty. In this diagram $\varphi_{min} = 45^\circ$ . The values in this diagram are not derived from calculations, only trial and error. A more thorough study into orientation optimization would likely yield in a different looking diagram. . . . .	32
5.13	Screen capture of the output from the orientation optimization algorithm. The top result has the lowest weight, and is in this case also grounded. . . . .	34
5.14	Orientation optimization of an antenna component. The result is grounded at the base of the antenna, but the bracket structure will need plenty of support if this product is to be printed in the orientation shown in (b). However, in (a) it would be necessary to use support structures inside and underneath the funnel, which would likely be more problematic. . . . .	34
5.15	The cylinder after being processed by the orientation optimization algorithm. The result is grounded with no problematic overhangs. . .	35
5.16	The arch-sphere benchmark model after run through both the orientation optimization algorithm, and the geometry alteration algorithm. This did not work as intended. . . . .	35
5.17	From left to right: The first image shows the unaltered original model. The second shows the same model, but the flat overhang has had an angle injected. The third shows how the single face algorithm now can take an advantage of the introduced angle, and thus pushes it to form a shape that does not need support. . . . .	36
A.1	This figure displays the ideal Gantt chart for the project. . . . .	II



# 1

## Background

The modern manufacturing industry relies on two types of processes: "forming" (such as casting and injection moulding) and "subtractive manufacturing". In subtractive manufacturing one piece of material is transformed into the desired shape by essentially subtracting material from it through for example carving or drilling. Additive manufacturing (AM) flips this on its head by instead incrementally adding material to a workpiece from the ground up. With this new technology an entire landscape of new opportunities has surfaced [1].

AM was recognized early for its potential as a prototyping tool in the product design phase. However, the technology has improved vastly over the years. This development has pushed AM from being merely a prototyping tool into becoming a means of manufacturing [1]. The combination of these developments and the possibility to "print" metal at low cost has attracted the attention of the space industry [2]. AM is a manufacturing technology that is well suited for the low production volumes typically associated with the space industry. As an added bonus, utilizing AM to build parts from the ground up would allow for the creation of designs that have less weight and a reduced amounts of interfaces [3].

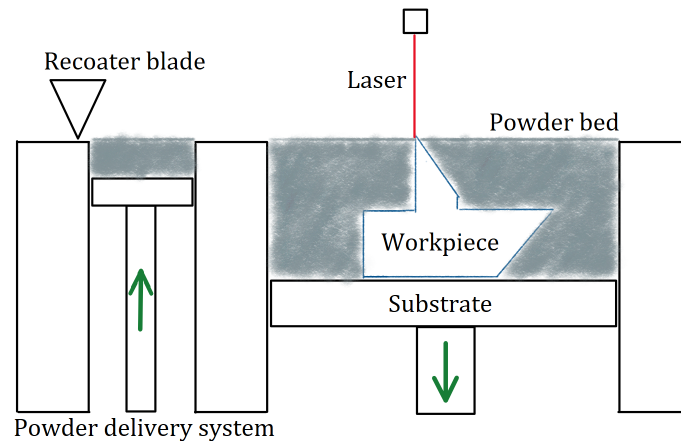
To fully embrace AM designers will need to learn how to create products in a new way. AM may prove beneficial to the space industry, but it does come with a new set of requirements and problems. One of those problems is the lack of tools to help designers who are used to designing for traditional manufacturing to transfer over to AM. As will be expanded upon later in this section, one particular type of tool appears to be missing in the designer's toolkit today. Specifically, a tool that automatically reshapes the geometry of models to make them more suited for AM, thus making the job of designers slightly easier.

### 1.1 A brief introduction to AM

Additive manufacturing is an industrial production technology which manufactures a product by building it layer by layer, from the bottom up. AM using metals can be done in many different ways. Three of the more common types are: powder feed, wire feed and powder bed fusion [4]. This thesis will focus primarily on laser powder bed fusion.

Laser powder bed fusion, also known as Selective Laser Melting (SLM) spreads a thin layer of powder over the workspace, which is then melted using a laser. After the

laser has finished working on the current layer, a new coating of powder is provided on top of the old one, and the process repeats itself until the component is done [1]. A sketch of a powder bed fusion system can be seen in figure 1.1. To transfer heat away from the workpiece "support structures" are used [5]. These support structures are usually created out of the same material as the component, and they also double as a physical support of overhanging geometries. Once the printing process is done the material quality of the product is close to that of its traditionally manufactured relative, rendering AM as a viable way to manufacture metal components [6].



**Figure 1.1:** Simplified sketch of a laser powder bed fusion system.

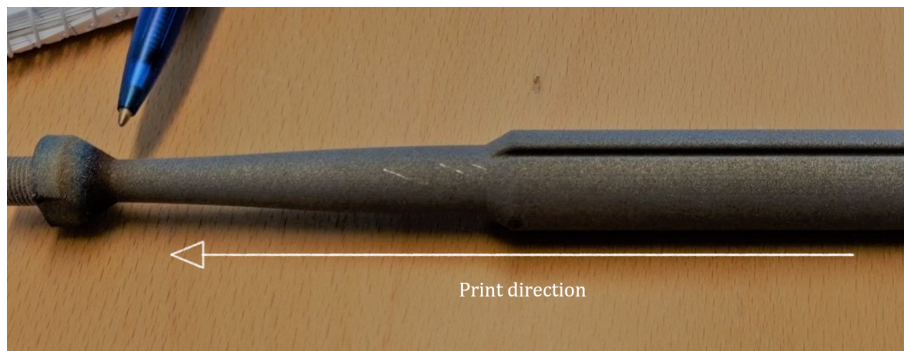
## 1.2 The space industry: a suiting candidate

The space industry currently relies a lot on casting, which has many downsides [3]. 1) Creating new casts is expensive. Making matters worse is the fact that things do not always go as planned. The casts might need to have their designs changed several times during a development cycle. When this happens the expenses grow, and resources are wasted. 2) Using casts results in long lead times. It takes time to design a new cast, and a lot of different factors need to be considered in order to avoid failure. 3) As a consequence to the previously mentioned problems, the practice of prototyping is reduced in the space industry. Rapid prototyping cycles in order to quickly test different designs can be an incredibly useful strategy to employ [7], but is less prevalent within the space industry. This reduces the willingness to attempt new designs, and so engineers often use old designs to minimize risk [8]. Leaning on old designs halts innovation. 4) Products within the space industry are generally not mass produced. The volumes are generally low. This means that traditional manufacturing can be quite cost-ineffective. However, some authors claims that additive manufacturing could potentially be able to counter some of these flaws [3].

As a result of utilizing AM casts are no longer required. This has two significant benefits: A new model can be created and printed with a significantly shorter lead time, and the large overhead costs for creating a new models shrinks. This opens up the possibility to utilize prototyping more often, and thus possibly makes innovation more likely [8].

Another large benefit for the space industry is the potential to reduce weight by using less materials. It has been demonstrated that the aerospace industry has a lot to gain from embracing AM, as less weight means less fuel [9]. This is especially true for more complex parts that are difficult to create using traditional manufacturing [10], in such cases the machining process itself becomes time consuming and costly, and so the benefits goes beyond mere materials saving. But in order to take advantage of any of these benefits all parts needs to be redesigned to utilize the inherent properties of AM as much as possible.

### 1.3 Problems



**Figure 1.2:** Area close to the top has darkened due to having close to insufficient heat dispersion. Could have been avoided by reducing the angle (increasing  $\varphi$ ), or by introducing support structures.

When manufacturing using AM some of the limitations of traditional manufacturing are alleviated, but are replaced with a new set of constraints. In order to design for AM a new way of thinking is required. Due to the heat produced during SLM thought must always be put in to how that heat should disperse. Not taking the heat dispersion into account may cause issues when printing overhanging features that lack proper support structures [6]. The powder itself does not transfer heat efficiently, and thus the heat builds up in the overhanging feature, resulting in poor surface quality [5]. An example of insufficient heat dissipation can be seen in figure 1.2. However, it is not as easy as simply inserting more support structures. The support structures themselves also result in reduced surface quality as they are directly attached to the surface of the product, and need to be removed manually after the printing process [6]. As a consequence of this the optimal solution would be to avoid the need for support structures as much as possible.

It can be challenging to embrace a new technology, and AM is no exception. To tackle this challenge there are two generic strategies available: 1) Design new products with AM in mind (otherwise known as Design for Additive Manufacturing or DfAM) [3], or 2) do not change design methodology, but make it compatible with AM during the finishing touches. While the second strategy might be more appealing to engineers who lack experience with AM, the first strategy will obviously

generate the best results. The problem is, as mentioned, the lack of experience and tools to make this strategy work [8].

### 1.4 The current toolkit

While designers lack a complete toolkit to aid in the creation of products compatible with AM, some tools have already emerged. In this section existing tools will be examined, as well as pointing out what the current toolkit is missing.

#### 1.4.1 Support structure generation

Inserting support structures manually can be a tedious and difficult task, and would likely lead to waste of materials and suboptimal results. To avoid this automatic support structure generators have been developed in a variety of ways [11]. An example of a software that can generate support structure for metal additive manufacturing is "Materialise Magics", which is a software developed by the company "Materialise" [12].

These tools do not only help by generating support structures, but they also help highlight problems for the designer. If an abundance of support structures are needed, then that might prompt the designer to rethink the geometry of the product. If used in this manner support structure generators can be used to acquire feedback on a products AM feasibility.

#### 1.4.2 Orientation optimization

Orientation optimization is used to find the best orientation in which the product can be printed. This is often done by rotating the product into different orientations, and evaluating mathematically which orientation is most suitable for AM [11]. Sometimes different orientations are given weights based on how suitable they are [13]. This tool is often used in concert with support structure generation to minimize waste of material.

#### 1.4.3 Automatic geometry alteration

Support structure generation and orientation optimization are relatively known tools among those who utilize AM. However, there is one type of tool which does not seem to be available to designers, specifically automatic geometry alteration (AGA). Research has been done on generating self supporting structures using voxels [14], but there appears to be a gap when the aim is to modify an existing geometry. An AGA tool directly modifies the geometry of the part to reduce (or completely remove) its need for support structures. A thorough patent search on "Google Patents" yields no relevant results when using terms such as "automatic geometry alteration" combined with "additive manufacturing" or "3D printing". This suggests that the idea of having an algorithm alter the geometric composition of a part is novel.



## 1.5 Impact on society

Improving manufacturing within the space industry does not come without its share of moral dilemmas. But, there are also reasons for why continuing to develop AM may have a positive impact on society.

### 1.5.1 The ethical and societal context

As it turns out our civilization might already have reached a point of no return when it comes to orbital debris. As of 2017 it has been reported that three to four satellites are lost every year due to collisions with orbital debris [15], and for each collision more debris is added [16]. This results in a "catch-22" scenario, we need to launch things in to orbit in order to learn, and improve technology. But, launching things into orbit might eventually prevent us from doing so in the future. However, if the quality could be improved of the things we launch into orbit, then we could also improve their lifespan. This is significant, because if the lifespan of satellites is prolonged, then the need for launching new satellites in order to replace old and broken satellites might be reduced.

There is also cause for concern related to military and security. AM can already be used to create weapon parts, and thus improving the technology could further emphasize this problem. Once organizations with malicious intent gains access to a printer and weapon schematics nothing can stop them from printing all sorts of devastating technology [17]. It could also be used by national military powers in order to create weapons and wage war.

Another important social aspect of the development of AM is its potential impact on the environment. In short, it might still be too early to say what effects AM will have. There are both positive and negative ways in which AM technology could interact with the industry to impact the environment. These are a few examples: 1) When creating components using AM less material is wasted. However, the material that is used can not be reused. 2) AM makes it easier to create light weight components, which could have a positive impact on the environment if for example used in aircraft to reduce the required amounts of fuel. This is especially important for the space industry, where weight always needs to be minimized. 3) AM might mitigate overproduction since products could be created on demand to a higher degree. This would reduce the need for large warehouses, and transportation [4, 18].

### 1.5.2 The potential impact of this project

This thesis project aims to in the end help designers create better products. If the end result of this project helps designers create more qualitative products, then it is possible that the lifespan of satellites can be increased. This could result in companies having less reasons to launch an excessive amount of man made objects into orbit. But, it might also result in making the technology more accessible and easy to use, resulting in an increase of launches to orbit. A similar duality can

often be found when discussing the potential usage of technology, as technology has always been a double-edged sword. Developing it often creates equal opportunities to help people, as well as to cause destruction. This is inevitable, but this particular technology (AGA) is unlikely to create any significant new threats. It could however make it easier to manufacture such technology. In summary, it is hard to tell how AGA could impact the looming issues with orbital debris and AMs potential for weapons manufacturing.

From an environmental perspective AGA could prove beneficial. Since one of the primary concerns is to reduce the required amount of support structures, then AGA might cut down the usage of materials during manufacturing, thus reducing waste. Additionally, by helping designers embrace AM low weight designs can be utilized to a broader extent. This could result in reduced fuel consumption during space operations.

Finally, there is at least one more potential problem that needs to be highlighted. When implementing automation of any task, a couple of questions ought to be considered: will the automation result in the loss of jobs? Or, will it merely result in alleviating tedious tasks from employees, allowing them to focus on other matters? This question can not be answered generally, but only on a case-by-case basis. Arguably it would be necessary to monitor the results of the geometry alteration software in order to keep quality under control. This would also provide a new task in the wake of the alleviated one. But, in order to reduce costs some companies may choose to rationalize away such quality controls. This would not only be a problem because of the loss of jobs. It would also be a questionable to put too much trust into an automatic algorithm, which if left unchecked might not always produce desirable results.

# 2

## Aim

The primary question that this thesis aims to answer is: *"How can the geometry of a product be automatically altered to reduce the need for support structures while maintaining product functionality?"*. To attain AGA a software was be developed, which was fitted into a graphical user interface.

### 2.1 Demarcations

In order to make this project feasible and focused two major demarcations will be set.

#### **Focus on geometric alteration**

This project will focus primarily on modifying the geometry of a part in order to reduce support structures. This also includes changing the orientation of the part in the printer to minimize the need for support structures. This project will not involve optimizing support structure type and density as that would make the scope of this project too wide.

#### **Optimize for laser powder bed fusion**

There are many different ways in which the additive manufacturing process can be performed. However, in this project focus will be to optimize support structures for SLM. There are two reasons for this: 1) Taking into account all different types of AM is too broad of a scope, and 2) Laser powder bed fusion is one of the main technologies in which the space industry is interested.

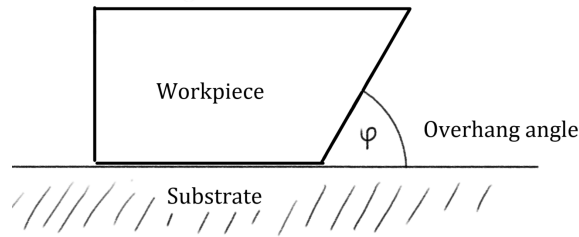
# 3

## Theory

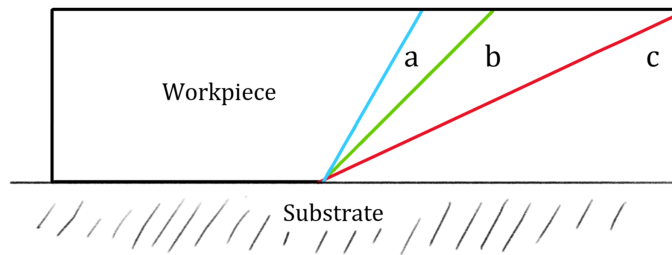
### 3.1 Metal additive manufacturing

In this section a few concepts surrounding the SLM process that are important in the context of this project will be brought up. At the heart of this section is the importance of overhang and support structures.

#### 3.1.1 Overhang



**Figure 3.1:** Illustration of the overhang angle, and how it relates to the workpiece and the substrate.

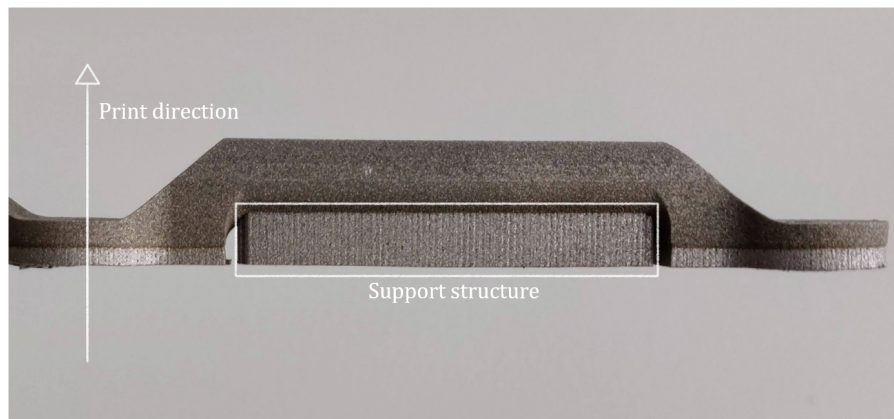


**Figure 3.2:** Illustration of three different overhang angles. a:  $60^\circ$ , b:  $45^\circ$ , c:  $25^\circ$ . Overhang a is stable, b is on the edge of stability and instability and c is potentially unstable.

Overhang describes a section of the workpiece that hangs above the substrate without direct support from underneath. Figure 3.1 demonstrates how this angle is measured. Depending on the sharpness of the angle there is more or less potential

for problems such as warping due to reduced heat dispersion, or in extreme cases collapse due to the lack of physical support [19]. A general rule of thumb is that an angle can be considered stable if it exceeds  $45^\circ$ , while angles smaller than that risk being problematic [20, 21]. In reality the exact angle may change depending on the material and laser scan speed [19]. See figure 3.2 for an example. Throughout the rest of the report, the overhang angle of a particular face will be referred to with the symbol  $\varphi$ , and the minimum allowed angle of a particular face will be referred to as  $\varphi_{min}$ .

### 3.1.2 Support structures



**Figure 3.3:** Support structure of a  $0^\circ$  overhang (also referred in this thesis as a "flat overhang"). Notice how the support structure is made of the same material as the part itself.

In order to get around the issues of overhang, designers can elect to utilize support structures. These structures help during the printing process both by providing physical support for overhanging features, and also a structure through which the heat can disperse. An example of a support structure can be seen in 3.3.

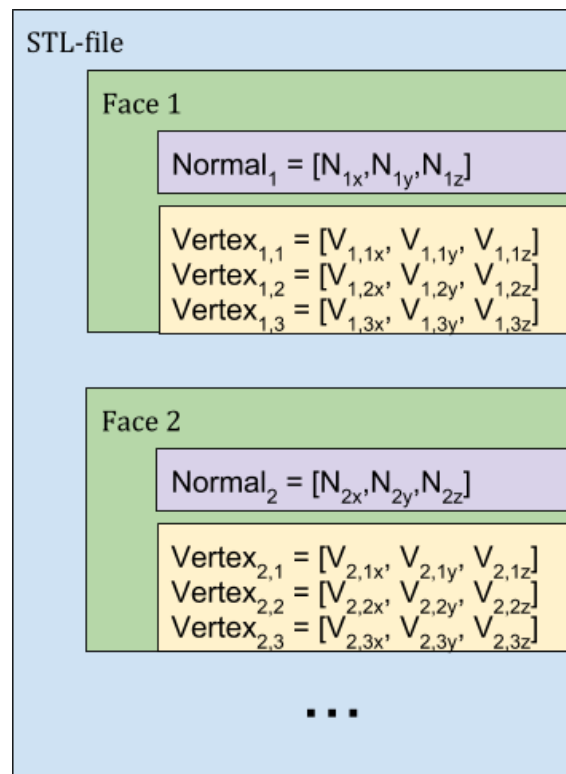
Unlike when printing in polymer materials, the support structures used when printing in metal are made out of the same material as the workpiece. This makes it relatively hard to remove the support structures, as they often need to be removed manually. This can have negative implications on several aspects of the manufacturing process [1]. Primarily, putting in time and resources to manually remove support structures from each printed product is inherently wasteful. One could also argue that since the support structures are removed they do not provide any value to the final product, and are thus a waste of material. In some cases it is not possible to remove the support structures due to their placement, which could potentially have detrimental effects on product performance (for instance, if the support structures were placed inside of a pipe used for transporting a fluid they would obstruct the flow).

## 3.2 Computer aided additive manufacturing

This section contains important concepts surrounding how computers can help with metal additive manufacturing.

### 3.2.1 The STL file format

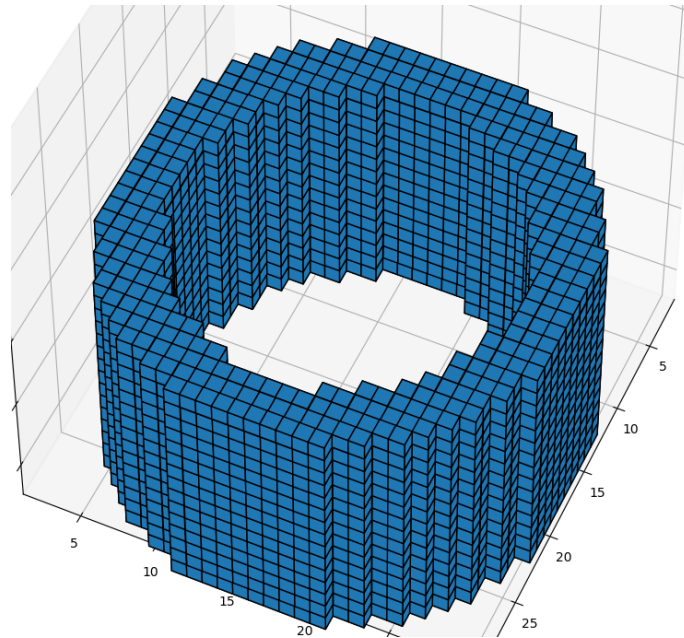
The .stl-file is a file format that is commonly utilized when 3D printing. It describes a three dimensional object by providing a set of faces. These faces are described using the coordinates of their respective vertices, and their normal vectors. One face consists of three vertices and one normal vector. It is also possible to extract the normal vector by utilizing the vertices to perform cross multiplication, as the vertices are always specified in a specific order to make such a calculation possible [22]. Figure 3.4 depicts a schematic of a typical STL file. STL files exists in three common variations: 1) The ASCII variation, which is a plain text format that is easy to read and parse, but results in large file sizes. 2) The binary variation, which is far more compact than its ASCII relative, and requires another parsing technique. 3) The colored binary variation, which has the exact same file structure as the regular binary variation, except it contains additional color information.



**Figure 3.4:** The structure of an STL-file. This example contains only two faces, but an STL-file could in theory contain any number of faces. Both binary and ASCII STL-files represents the data using similar architectures (the binary version contains slightly more information which is not relevant to understanding how the file is structured). The colored variant contains slightly more information that is not relevant for this thesis.

### 3.2.2 Voxel representation

Voxel representation utilizes a three-dimensional matrix where each element is a boolean value (true or false, often represented by a one and a zero). Each element represents a limited amount of space in the actual object. If the object possesses matter in a certain point in space, then that corresponding point is set to "true" in the matrix. See figure 3.5 for an example of a voxelized cylinder. The process of converting an existing geometry to a voxel representation of the same object is referred to as "voxelization" [23]. Using this type of representation for a model often requires large amounts of memory due to how large the matrices tend to become, especially at higher resolutions. Processing such large matrices can be very resource heavy and time consuming.



**Figure 3.5:** A voxelized cylinder. Each cube represents a voxel. The entire model is a representation of a  $30^3$  matrix consisting of ones and zeros. Each element that contains the number "one" is drawn using a cube, while all the other elements that contain zero remain unrendered.

# 4

## Method

There needs to be a thorough understanding of what types of geometries causes problems, and how such problems can be mitigated or countered. To create such an understanding a literature study was conducted, which formed the basis for the introduction and theory sections of this thesis. The conclusions from the literature study were used as a guideline during the development of the software. On top of the literature study two interviews were conducted. The first interview was with an expert in the subject of metal additive manufacturing, and the second with a manager from within the space industry. An initial time plan was created, which can be seen in appendix A.

### 4.1 Interviews

During this project two interviews were performed. The first interview was held with a researcher at Chalmers, who is an expert on the subject of metal additive manufacturing. The interview was semi structured in order to extract as much information as possible. These were the the topics of discussion:

- Quality issues associated with support structures
- Importance of model orientation
- Expectations from a geometry altering software

The second interview was held with a manager at a Swedish aerospace technology company. This company manufactures satellite components, and recently started exploring additive manufacturing. This interview mainly focused on how geometry alteration could be of use to them, which helped steer the development of the software in the right direction.

### 4.2 Benchmarks

In order to measure success when testing the software a set of benchmark models were created. The benchmark models provide a variety of challenging features, and also range in model complexity. These benchmarks makes it possible to find issues with an algorithm, as well as its potential strengths. All benchmarks are listed together with depicting figures in appendix B.



### 4.3 Concept generation and selection

For concept synthesis the method of choice was brainstorming. Using this technique a set of concepts were created both for the problem detection algorithm, and the problem correction algorithm. When selecting a concept for problem detection a thorough evaluation of each concept was conducted in order to find which concept best suited the needs of this project, and eliminate the others. When selecting a concept for geometry alteration (problem correction) a weighted Pugh matrix was utilized [24]. Instead of having a customer provide requirements for the final product all criteria were derived from problems that were uncovered during the literature study. On top of the criteria that were derived from the literature study three other criteria were put into place as a consequence of the research question, the resources available during the course of this project, and the project time limit. These criteria were:

1. The concepts should maintain product shape and functionality.
2. The concept should be able to be run on an average modern computer (referred to as processing time).
3. The concept should be realizable during the set time of this thesis project (referred to as complexity to construct).

The first criteria is a direct consequence of the research question, which specifically states that product functionality needs to be maintained. Criteria two was a necessity due to the computing power that was available during this project, this is synonymous to minimizing processing time. Criteria three handles potential knowledge gaps in the concept, and the time it would take to close this gap. This criteria is derived from the thesis project time limit.

### 4.4 Creating the prototype

The programming language of choice was Python. The reasoning behind this is that Python has a vast mathematical library, similar to Matlab. The reason Python was selected rather than Matlab is because Python is free, flexible, easy to integrate into other applications, can be converted into a single executable program, and comes with the possibility of creating a GUI to ease interfacing with the script which was deemed necessary for the creation of a presentable prototype.

The development itself was divided into two stages. The first stage was the creation of a problem detection algorithm which needed to be able to identify problematic angles, and provide information about where in the model they exist. The second stage was to create a geometry alteration algorithm, also referred to as a problem correction algorithm. The purpose of the problem correction algorithm is to fix the problems detected by the algorithm created in the first stage.

# 5

## Results

### 5.1 Interviews

The interviews provided some valuable insight into how AM is to be used in the space industry, and also some problems with the existing tools. This section will cover two important pieces of information that were uncovered during interviews, which came to be of high importance.

#### 5.1.1 The problem with orientation optimization

During the interview with the researcher at Chalmers, some important points regarding orientation optimization was brought up. Software that automatically orients the model to find an optimal printing orientation can be used to minimize support structures, but there are some common issues. An important issue is that the support structures that are generated in the new orientation may be hard to remove. It is often preferable to remove support structures from flat surfaces, for such surfaces are easier to process (for example, when milling). The researcher also mentioned that they do not take heat dispersion into account, which may cause problems during printing.

#### 5.1.2 Strict requirements in space technology

When interviewing a manager from within the space industry manufacturing branch an important detail was brought up: some surfaces may not be altered. For example: any surface that compose the inside of a waveguide, or surfaces that influence the geometry of interfaces with other parts. Some of these surfaces should also be avoided when placing support structures. For example, if an orientation of a waveguide requires internal support structures which can not be machined away then that would make the waveguide useless. This suggests the need for a way for the user to select important surfaces on the model which under no circumstances are to be touched, either by support structures or geometry altering algorithms.

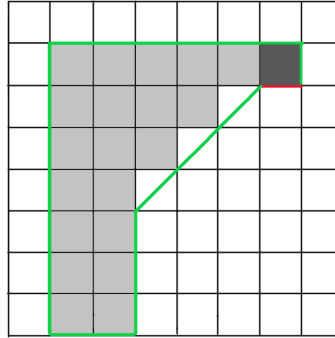
### 5.2 Problem detection algorithm

Before starting the development of the software a brainstorming session was held to generate ideas for how problems could be detected in an STL-model. The session resulted in three different strategies, which will be presented in the following section.

### 5.2.1 Generated problem detection concepts

These are the concepts that were generated for detecting problems in an STL geometry.

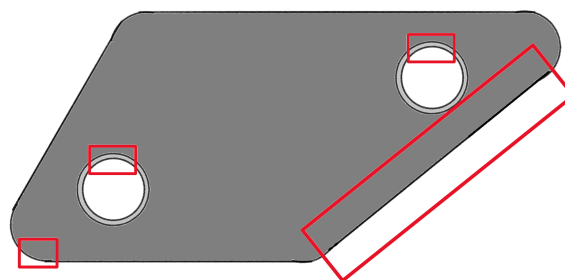
#### Voxel parsing strategy



**Figure 5.1:** 2-dimensional illustration of a voxelized structure. The original structure is represented by the green (supported faces) and the red (unsupported faces) contour. The greyed out grid elements represents the voxel representation of the original structure. The dark grey voxel lacks proper support, as it does not have an adjacent voxel directly or diagonally underneath it.

This strategy would implicate examining each individual voxel to determine whether or not that voxel is supported. This could be done merely by checking if an activated voxel has another activated voxel adjacent to it in the layer directly underneath. See figure 5.1 for an example. Utilizing voxel parsing is likely to be taxing for the system memory due to the large matrices involved. However, it does have the benefit of being a rather simple and easy to understand approach.

#### Machine learning strategy



**Figure 5.2:** Concept of how machine learning might be able to be used to recognize problematic areas of a model using image recognition. The areas highlighted with the color red would be considered as problematic areas, identified by the machine learning algorithm.

An image recognition algorithm based on machine learning could be used to identify problematic areas of an object, a concept of which can be seen in figure 5.2. Several

images of cross sections from the object could be used. Those images could possibly be divided into a grid, to make it easier to extract the coordinates of any problematic areas. In that case each element of that grid would be examined by the algorithm. If the algorithm finds a problem in a given grid element then that specific element could be translated into coordinates in the CAD-model, thus pointing out the error on the actual model.

### **Normal vector analysis strategy**

The STL-file format contains all the information needed to calculate potentially bad overhang angles. Thus, an algorithm could parse through all of the faces stored in the STL-file, retrieve their normal vectors and compare that normal vector to the Z-unit vector. In this way the overhang angle  $\varphi$  can be calculated for each individual face. If the angle is underneath a specified threshold value then that particular face can be flagged as a problem.

### **5.2.2 Evaluation of concepts**

After the creation of the strategies they were all evaluated in detail. In this section the potential pros and cons of each strategy is explored.

#### **Voxel parsing strategy**

Voxelization was attempted. However, the process of converting the STL-file to a voxel matrix with an acceptable resolution proved to be very time consuming. Which brings up another problem with voxelizing: it requires some amount of user interaction in order to set the resolution (voxel density), which might require several attempts before an adequate result is yielded. A high resolution is always better, but there is a significant trade-off between resolution and performance, as high resolution voxel matrices puts a lot of strain on system memory. Compared to the other methods, this approach requires the most computer resources, and takes the longest time to process. It also becomes complicated to make  $\varphi_{min}$  variable using this approach, but relatively straight forward when  $\varphi_{min} = 45^\circ$ . Because of the invariability of  $\varphi_{min}$  and the long processing time this strategy was discarded.

#### **Machine learning strategy**

The machine learning approach was dismissed for several reasons. Primarily, compared to the other methods this is by far the most advanced to develop. The machine learning strategy would also require a large data set in order to reliably be able to classify a geometry as problematic or not. It would also be problematic to make the minimum overhang angle ( $\varphi_{min}$ ) variable. Making  $\varphi_{min}$  variable could be done using separate training data sets with different values of  $\varphi_{min}$ , but that would put further emphasis on the problem that such data sets are not available for use in this project, and would take too long to create.

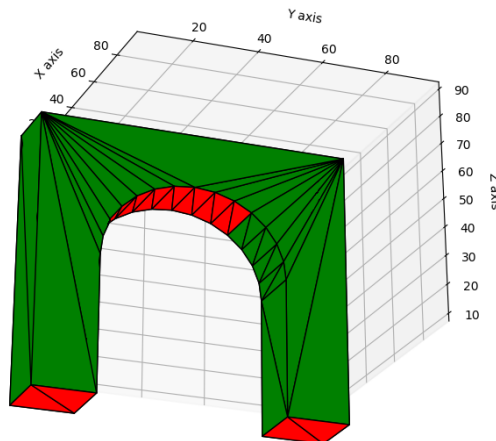
#### **Normal vector analysis strategy**

It is possible to detect problematic overhangs using the information that exists inside the STL file format. By extracting the normal vectors of each face the difference between the normal vector and the negative Z-unit vector can be used to determine

whether or not a face has a problematic overhang. Since the STL-format contains all the necessary information to conduct such an analysis the normal vector analysis works without issues on both basic and complex geometries. It is also a very fast method since it does not require the model to be converted into voxels (as in the voxel parsing strategy) or a set of images (as proposed in the machine learning strategy).

### 5.2.3 Concept selection and result

The results of the investigation of problem detection strategies resulted in the choice of developing a normal vector analysis algorithm. Primarily because of its ability to detect problems in all kinds of geometries no matter the complexity, but also because of how quickly it could do so. Figure 5.3 displays the first version of problem detection using normal vector analysis. This early version did not take into account that faces which touch the ground does not require support, as they are already supported directly by the substrate. When the detection algorithm was integrated into the final prototype this issue was fixed. This was done by identifying the ground level by looking for the lowest Z-index in the model. Any vertices that share the lowest Z-value that is associated with the ground level must be touching the ground/substrate.



**Figure 5.3:** The first successful problem detection experiment. The Z-direction represents the printing direction. The red faces represent surfaces that possesses an overhang angle that is too small (in this example too small is considered to be  $\varphi < 45^\circ$ ). The green surfaces are problem free. This early in development there was no regard for surfaces that touched the ground, which has resulted in the lower "feet" also being registered as problematic surfaces, despite them being supported by the substrate (the substrate is not visible in this figure).

## 5.3 Basic problem correction algorithm

In this section a way of correcting the detected problems will be explored. Somehow the geometry is to be altered to reduce or remove the overhang surfaces. Again, a

set of strategies were generated through brainstorming.

### 5.3.1 Generated correction concepts

These are the concepts that were generated for altering the geometry of an STL model.

#### **Voxel support strategy**

This strategy could be seen as an extension of the voxel parsing problem detection strategy. Once an unsupported voxel is detected, then the empty space beneath that unsupported voxel is filled, in order to support it. Again, this has the benefit of being a clear and easy to understand approach. However, it also inherits the flaws of the voxel parsing strategy, meaning that it requires a lot of computing resources and time for such a method to be applied to an actual model.

#### **Hybrid strategy**

This approach serves as a hybrid between handling the problem using vectors, and handling it using voxels. The idea is to first run the normal vector analysis problem detection, and then use the information gathered to run a focused voxel support algorithm. This means that only the voxels relevant to the problematic overhangs would be handled by the algorithm, which would likely result in a swifter processing time. This could result in a faster overall performance than the first mentioned voxel support strategy, but it would likely be much more complicated to develop.

#### **Vertex manipulation strategy**

The vertex manipulation strategy aims to utilize the STL-format as much as possible. Since the STL-format contains all faces, normal vectors and vertices, then it could be possible to directly alter the position of the vertices in order to correct any problematic overhang angles. Using the normal vector analysis strategy for problem detection it should be possible to directly identify which vertices should be targeted for manipulation.

### 5.3.2 Concept evaluation and selection

Before development of a problem correction algorithm could start, a concept needed to be selected. To aid in this process a Pugh matrix was utilized which can be seen in table 5.1. This tool helped create an understanding of how the different solutions might perform from the perspective of the different requirements. During this evaluation the voxel parsing strategy will be referred to as VP, the vertex manipulation strategy as VM, and the Hybrid will keep its title as "Hybrid".

Since this product development projects does not have a customer from which to extract requirements, the requirements were instead based on feasibility, how well different concepts solves the core problem and if the tool is in fact usable. Each criteria has its own weight, which represents how important it is. Each concept is "scored" in each separate criteria with one of three different scores: -1, 0 or 1. A score of "-1" means that performance for the specified criteria is expected to be bad.

A score of "0" means that the performance is unknown or mediocre. A score of "1" translates to the concept having the potential of performing well.

Concept selection pugh matrix				
Criteria	Weight(1-3)	VP	VM	Hybrid
<b>Solves core problem</b>				
Eliminates problematic angles	3	1	1	1
Maximum overhang angle is variable	2	0	1	1
Maintains product shape/functionality	3	0	0	0
<b>Usability</b>				
Processing speed	2	-1	1	0
Utilises the STL-format as input and output	3	0	1	0
<b>Feasibility</b>				
Complexity to construct	1	1	-1	-1
<b>Score</b>				
Sum		1	3	1
Weighted sum		2	9	4

**Table 5.1:** A Pugh matrix used to aid in selecting a suitable problem correction concept to develop. The three concepts are "Voxel parsing" (VP), "Voxel Manipulation" (VM) and the Hybrid.

### Solves core problem

Problematic overhang angles are the prime issue which needs to be solved, and its place on the list of criteria is thus self explanatory. At the time when these concepts were conceived it was thought that they all possessed the capacity to eliminate problematic angles. For this reason they all score full points in the "Eliminates problematic angles" category.

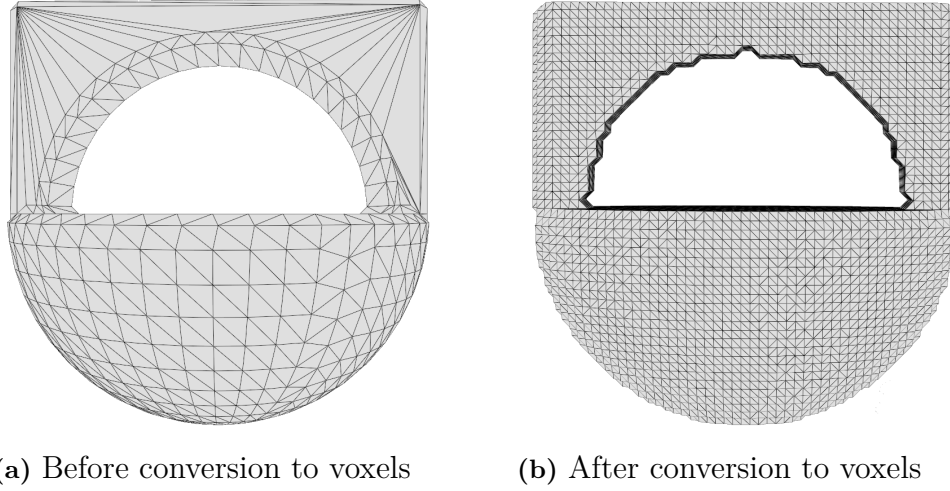
Different materials and different circumstances may warrant a variety of values for the minimum overhang angle. When looking at the variability of  $\varphi_{min}$  then VP scores lower because it is harder to handle small adjustments to angles using only voxels, while the VM and Hybrid concept both treat angles using vectors, which is much easier.

The product functionality must remain intact after all problematic angles have been dealt with. However, it is impossible to make a blanket statement for any of the concepts regarding whether or not that concept will maintain product functionality. For this reason all concepts receive no points in this category as it can only be judged on a case-by-case basis.

### Usability

The first category under "Usability" treats processing speed. This category is important because a method that takes too long will likely never be used. VP scores the lowest because of how time consuming parsing voxels can be. The hybrid comes next because it too utilizes voxels, but possibly in a more efficient manner. VM gets

the highest score since it only utilizes vector and matrix operations, which is likely to take the least amount of time.



**Figure 5.4:** Voxel conversion information loss. Figure **a)** is an image of the original STL-file containing the arch-sphere benchmark. Figure **b)** shows a version of the arch-sphere that has been converted to a voxel format, and then back into STL again. The visibly jagged edges, and the increased face density are results of the imperfect conversion process.

The next category touches upon interoperability. Most CAD-software can import and export STL-files, which is a widely used format when utilizing any sort of 3D-printing technology. Therefor the concept should be able to import and export to STL without any issues to prevent complications to the work flow of any designer who is using it. In this category VP and Hybrid scores lower than VM. However, this is not because they can't import or export from/to STL, but because they need to at some point convert STL to voxels, and back again. This conversion of data structure can cause loss of information (see figure 5.4a for a visible example), and other issues.

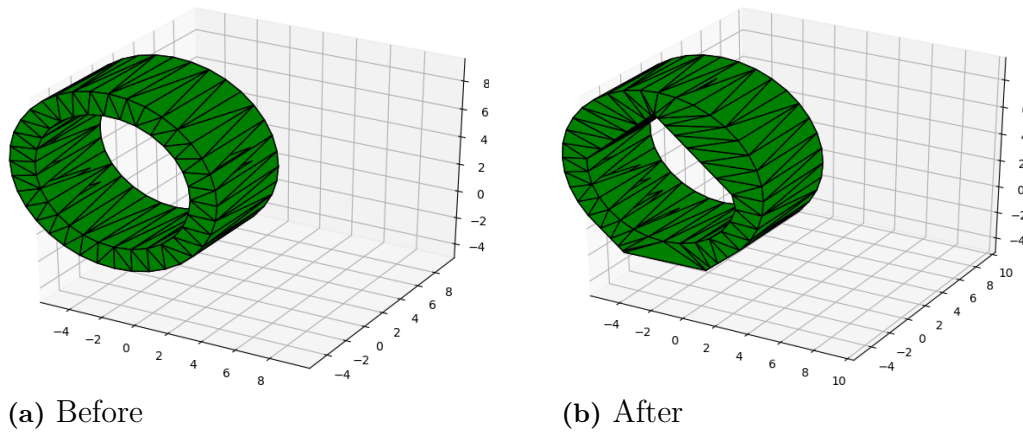
### Feasibility

Finally, the Pugh matrix touches upon the issue of complexity, which was an estimation of how hard it would be to realize the concepts. VP appears to be the easiest to implement due to its straight forward "brute force" nature, and thus scores the highest in this category. VM and the Hybrid both receive a lower score due to their high complexity and knowledge gaps as it is not clear how any of these two methods would work in detail.

### Final selection

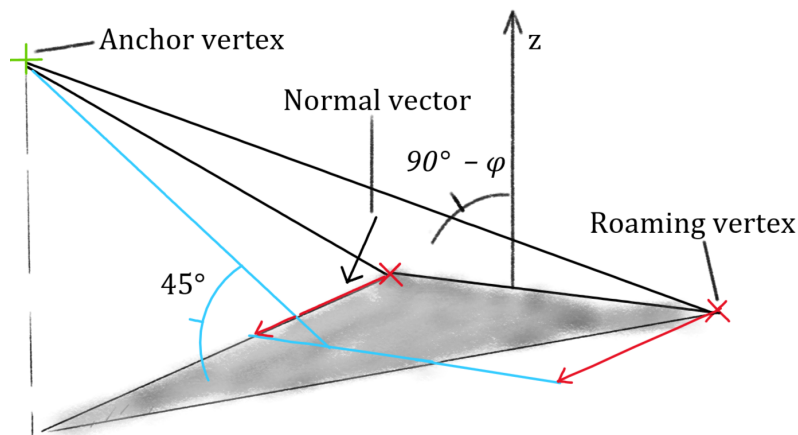
Vertex Manipulation aggregated the highest score, and was thus selected for further development. Early tests suggested that there was some merit to this solution, as can be seen in figure 5.5 which depicts the result of one of the benchmark models being run through an early version of the algorithm.





**Figure 5.5:** On display is the cylinder benchmark, before and after it passed through the problem correction algorithm. In **b)** the model is shown after it has passed through the algorithm, which has evened out the bottom section and given the top half of the inside of the cylinder a  $45^\circ$  slope.

### 5.3.3 Focus on one face at a time



**Figure 5.6:** This figure depicts the single face algorithm (SFA). Seen in the figure is a single face, and its components: the two roaming vertices, the anchor vertex, and the normal vector. The two red arrows represents the change vectors, which are calculated using the SFA. The change vectors always points in the same direction as the normal vectors projection on the XY-plane, and are later used to resolve problematic overhang angles. In this example the desired minimum overhang is  $45^\circ$ . Thus, the change vectors strive to push the roaming vertices in a manner that would result in a  $45^\circ$  overhang.

In order to develop a comprehensive geometry alteration algorithm it was decided that the algorithm should focus on one face at a time. That way, it would be easy to understand and debug. The SFA does exactly that. It accesses one single face from the STL-file and attempts to correct the problematic angle, if it has one. Here is how it works:

**Step 1:** Gather all of the vertices of the face, and sort them by their respective Z-index. If they all have the same Z-index, then  $\varphi = 0^\circ$  in which case the algorithm stops.

**Step 2:** Mark the highest vertex as the "anchor" vertex. The anchor vertex will remain stationary throughout the rest of the algorithm. The other two vertices are considered "roaming" vertices as long as they do not share Z-index with the anchor. The purpose of a roaming vertex is to eventually move in order to increase  $\varphi$ . An example of how this might look can be seen in figure 5.6.

**Step 3:** For each roaming vertex, calculate how far that vertex needs to move in order to increase  $\varphi$  to  $\varphi_{min}$  (which by default is  $45^\circ$ ). In this calculation, the original normal vector of the face is used. This is of high importance. The normal vector is used as a guide in this calculation in order to determine the direction in which to push the vertices. The reason the original normal vector is used is because it helps maintain the shape of the model even after it has passed through the algorithm a few times. The calculation is performed as such:

$$\begin{aligned} t_{xy} &= \frac{\vec{v}_{a,z} - \vec{v}_{r,z}}{\tan(\varphi_{min})} \\ \vec{n}_{xy} &= [n_x, n_y, 0] \\ \hat{n}_{xy} &= \frac{n_{xy}}{|n_{xy}|} \\ \vec{w} &= \vec{v}_a - \vec{v}_r \\ |\vec{w}_{xy}| &= \vec{w} \cdot \hat{n}_{xy} \\ d &= |\vec{w}_{xy}| - t_{xy} \\ v^* &= d \cdot \hat{n}_{xy} \end{aligned}$$

where

- $t_{xy}$  is how far the roaming vertex should be from the anchor in the xy-plane
- $\vec{v}_a$  is the coordinate vector of the anchor vertex
- $\vec{v}_r$  is the coordinate vector of the roaming vertex
- $\varphi_{min}$  is the smallest allowed overhang angle
- $\vec{n}$  is the original normal vector of the face.
- $d$  is the distance  $v_r$  needs to be pushed in the  $\hat{n}_{xy}$ -direction to correct the angle.
- $v^*$  is the change vector, which later will be used to alter  $v_r$ .

**Step 4:** Store in memory that the roaming vertex needs to have its coordinates changed by  $v^*$  to correct the angle for this face. The algorithm is now done.

### 5.3.4 Performing the change

The reason why  $v^*$  is not directly used to change the position of the individual roaming vertices in the last step of the algorithm is because several faces may share the same vertex. Thus, if a vertex is moved to correct one face, it could potentially alter the state of another adjacent face and cause new problems. To counter this,

all values of  $v^*$  that are calculated for each face are stored in memory together with a relation to their respective vertices. Once the SFA has been executed for each face in the model, another loop is initiated. This loop runs through each vertex in the model. The loop calculates the mean change vector for each vertex, as such:

$$\bar{v}^* = \frac{\sum_{i=1}^N v_i^*}{N}$$

where

- $\bar{v}^*$  is the mean change vector for a particular vertex
- $N$  is the amount of change vectors stored for a particular vertex
- $v^*$  is a change vector

After the mean change vector ( $\bar{v}^*$ ) has been calculated it is simply added to the current coordinate vector of the vertex, thus changing the position of the vertex:

$$v_{new} = v_{old} + \bar{v}^*$$

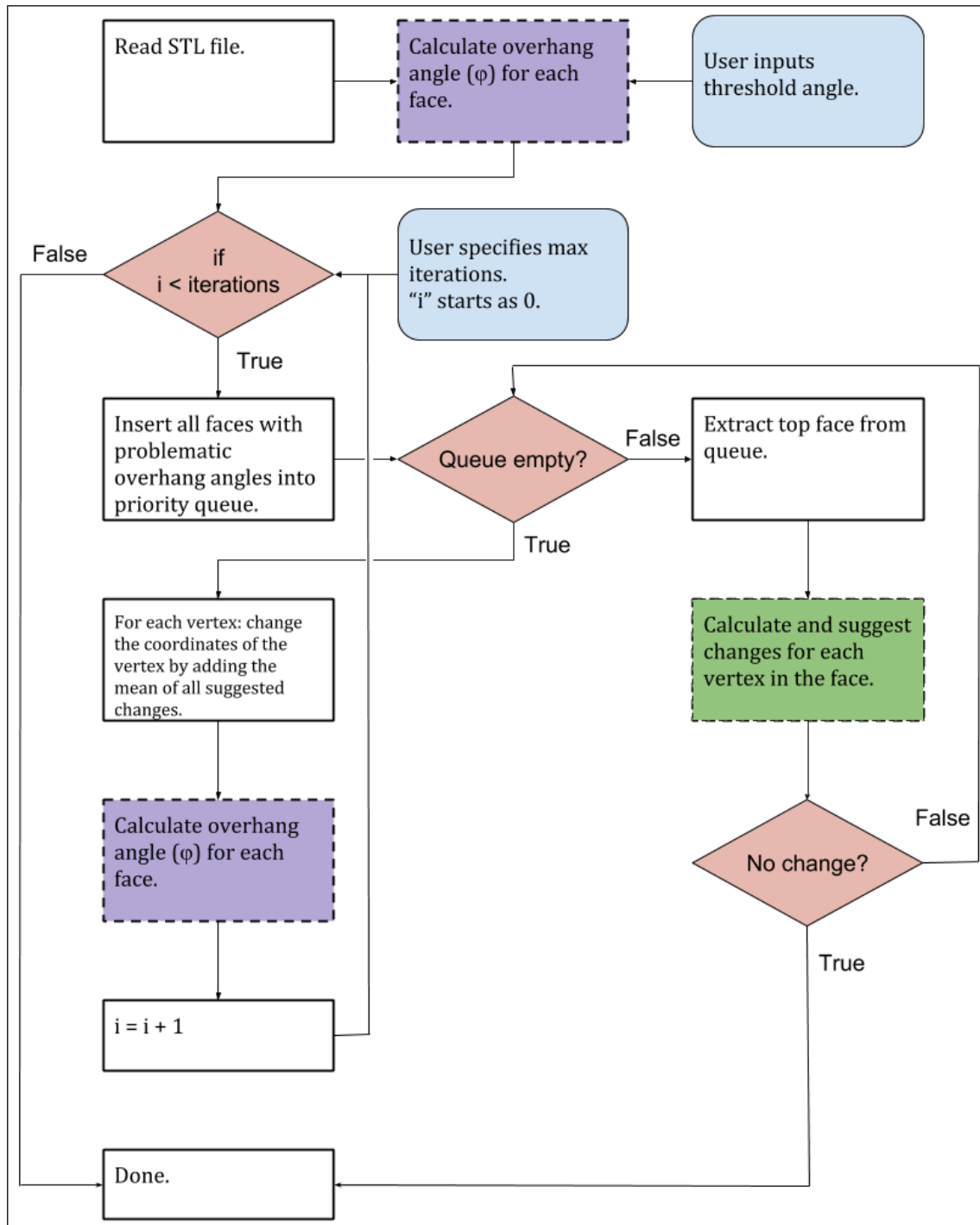
where

- $v_{new}$  is the new updated vertex position vector
- $v_{old}$  is the old vertex position vector
- $\bar{v}^*$  is the mean change vector.

### 5.3.5 The meta algorithm

In order to tie everything together, there is a meta algorithm which determines when to do what. The meta algorithm is illustrated in figure 5.7. The meta algorithm starts off by reading whatever STL-file the user wishes to process. It then searches for faces with problematic overhang angles (what constitutes a problematic angle can be specified by the user, but it defaults to  $45^\circ$ ). The algorithm then enters a loop where it calculates change vectors for the vertices inside each problematic face using SFA (see 5.3.3). Once it has performed this calculation for each problematic face it applies all of the suggested changes at the same time using the method described in 5.3.4. Once the change vectors have been applied, the problem detection algorithm is repeated, and the loop starts over. The loop keeps running until one of three situations occur:

- The loop has reached its max amount of iterations.
- The problem detection algorithm fails to find any more problematic faces, thus there is nothing left to correct.
- The amount of problematic faces has stopped changing after  $n$  iterations, where  $n$  is a number defined by the user. This is referred to as convergence.



**Figure 5.7:** The meta algorithm. This algorithm shows how the detection algorithm (purple) works together with the correction algorithm (green) in order to produce results. The queue does not keep the faces in any particular order.

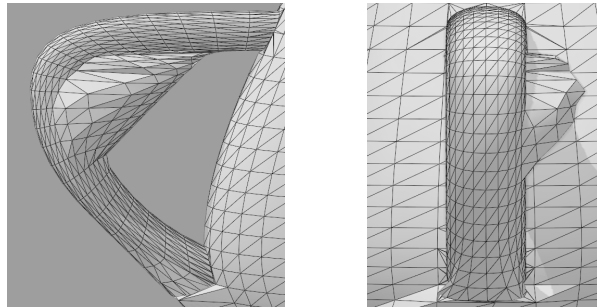
### 5.3.6 Problems with the basic problem correction

The problems with the algorithm can be divided into two categories. The first category of problems affect how the algorithm operates. The second category is performance, which translates to speed and reliability.

#### Operational problems

The basic problem correction algorithm as presented thus far has three known operational issues:

1. It can't handle faces with an overhang angle of  $0^\circ$  ( $\varphi = 0^\circ$ ).
2. It often has problems with curved geometry.
  - (a) Swelling or "blobbing" around high density curved surfaces. See figure 5.8.
  - (b) Inability to provide a solution for curved surfaces that touch the ground
3. Since the SFA focuses on one face at a time, it does not take into account if the change vectors it proposes will cause a collision with another face. This can result in invalid geometries.



**Figure 5.8:** "Blobbing" on a surface with a high face density, seen from two different perspectives. This particular model is of the handle of a teapot.

In order to take on these issues the focus needed to be shifted away from solely looking at one face at a time. Evidently, a few other components besides the SFA were required to make the problem correction algorithm more versatile.

#### Performance problems

One major problem which relates to both speed and reliability is the issue of vertex identity. One of the core routines of the algorithm is to identify each unique vertex. Checking if two vertices are the same could be as simple as checking the equality of each of the coordinate components (eg.  $x_1 = x_2, y_1 = y_2, z_1 = z_2$ , where the indices 1 and 2 represent two different vertices). However, due to how different CAD software exports to STL there is sometimes a slight difference between how two separate adjacent faces represent the same vertices. Thus, if the algorithm checks for equality of each coordinate component, then two vertices that are actually the same might be considered to be different by the program. This is referred to as a leak, meaning that the surface of the model is not closed. Leaks can cause some models to be ripped apart in the problem correction process, rendering the process unreliable.

## 5.4 Advanced problem correction algorithm

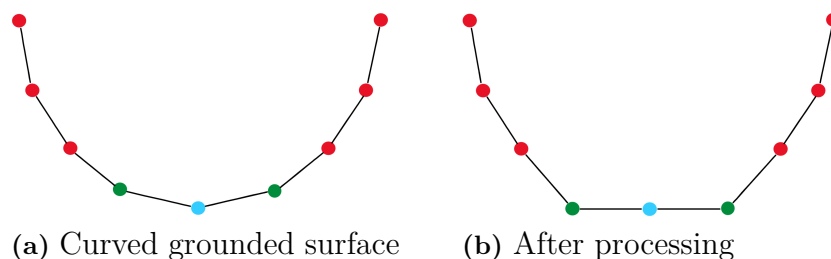
The advanced problem correction algorithm aims to provide solutions to the issues with the basic problem correction algorithm. This section will bring up possible solutions for the problems that were encountered when developing the basic correction algorithm (see section 5.3.6), and why some problems could not be solved.

### 5.4.1 Addressing the problems with curved geometries

Curved surfaces sometimes resulted in unexpected consequences when running the basic problem correction software. Two of the most prominent issues that were encountered was issues with curved surfaces that touched the ground, and a disfigurement caused by the algorithm malfunctioning, referred to as "blobbing".

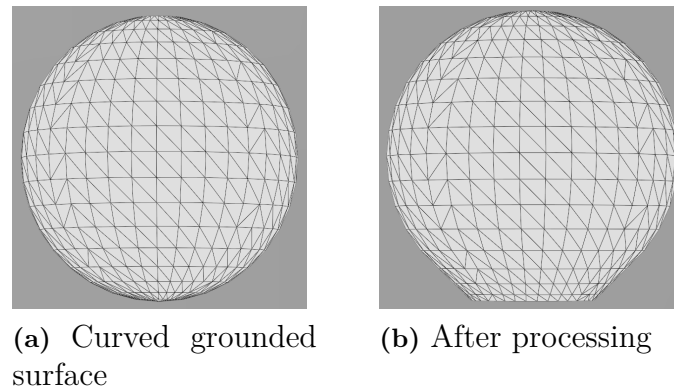
#### Grounded curved surfaces

One problem which could be fixed in the advanced algorithm was the problem with curved surfaces that touch the ground. As mentioned in the interview with the researcher at Chalmers (see 5.1.1), it is often preferable to remove support structures from flat surfaces. With this in mind it was possible to develop an addition to the meta algorithm which deals with rounded surfaces that touch the ground.



**Figure 5.9:** Rounded surface flattening. If a vertex touches the ground, and the surrounding vertices does not, then the surrounding vertices are brought down to ground level. This could reduce both the need for support structures and the need for machining in post processing. The bright blue dot represents the vertex that is touching the ground, and the green dots represent the surrounding vertices.

In order to treat rounded surfaces that touch the ground the algorithm looks for one specific property: a vertex that touches the ground, while all surrounding vertices does not. An illustration of such a situation can be seen in figure 5.9a. Once a vertex which fits that description is encountered the algorithm "flattens" the vertices surrounding it, thus creating a flat base from which the rest of the model can be printed from (see figure 5.9b). By providing such a base the need for support structures in the surrounding area may be mitigated or resolved. It is important to consider heat dissipation when moving from a narrow geometry to a wide geometry, so while this technique might reduce the need for support structures it might not remove it completely.



**Figure 5.10:** This figure displays the algorithm in use. In this case the algorithm has detected a curved surface where only one vertex touches the ground. It has thus flattened the surrounding vertices, reducing the need for support structures. After flattening the bottom, the SFA pushed out the vertices at the bottom, creating a stable  $45^\circ$  overhang.

## Blobbing

This problem seems to be caused by how the direction of pushing vertices is defined. As described in 5.3.3, the SFA only pushes vertices in the direction of the normal vector projected onto the XY-plane. This seems to sometimes create these "blobs" on curved surfaces with a high face density. This problem could not be solved within the time frame of this project.

### 5.4.2 Dealing with leaks

In order to deal with leaks the method for controlling vertex equality was changed. Rather than comparing the individual coordinates of each vertex, the distance between the vertices is measured. If the distance falls within a tolerance threshold, the vertices are considered to be equal. To ensure that there are no leaks present, the algorithm then gathers each edge that exists within the model. Once all edges have been gathered, if any edge does not belong to exactly two faces, then there must be a leak. If such a leak is encountered, the user is prompted. To counter the leaks, the user can then choose to increase the vertex equality threshold, making vertices that are further from each other count as equal, thus reducing the probability of leaks.

### 5.4.3 Overhang without an angle

A major problem with the SFA is its inability to deal with overhangs where the normal vector lacks an angle. From now on such overhang surfaces will be referred to as "flat overhangs". Since the SFA pushes vertices based on the projection of the normal vector onto the XY-plane, then a normal vector that is perpendicular to the XY-plane would result in the vertices not being pushed at all. This renders the SFA completely inapplicable to flat overhangs. To deal with this problem a handful of ideas were generated.

### **Introduce an angle**

It might be possible to force an angle upon a flat overhang, thus making it possible to apply the SFA onto it. However, this solution breeds new problems. Mainly, in which direction should the flat overhang be tilted? Tilting it in an improper direction might have devastating implications on product functionality.

### **Merge**

In some cases it might be possible to merge the edges of a flat overhang, which would eliminate that surface completely. But similar to introducing an angle, this proposition requires the algorithm to figure out which edges ought to be merged in order to preserve functionality. It also becomes useless if the flat overhang has an uneven amount of edges, such as a triangularly shaped flat overhang. In such cases it may be possible to merge all vertices instead.

### **Orient model to avoid flat overhangs**

An orientation optimization algorithm could be developed which penalizes any orientation which has flat overhangs. This would not actually solve the problem, but it would provide a way to move around it.

### **Utilize voxels on flat overhangs**

Once the SFA has finished running and can no longer find any edges to solve the model could be converted into voxels. Then, using voxel problem correction the flat overhangs could be eliminated by either adding or removing voxels. This again has issues related to conservation of functionality.

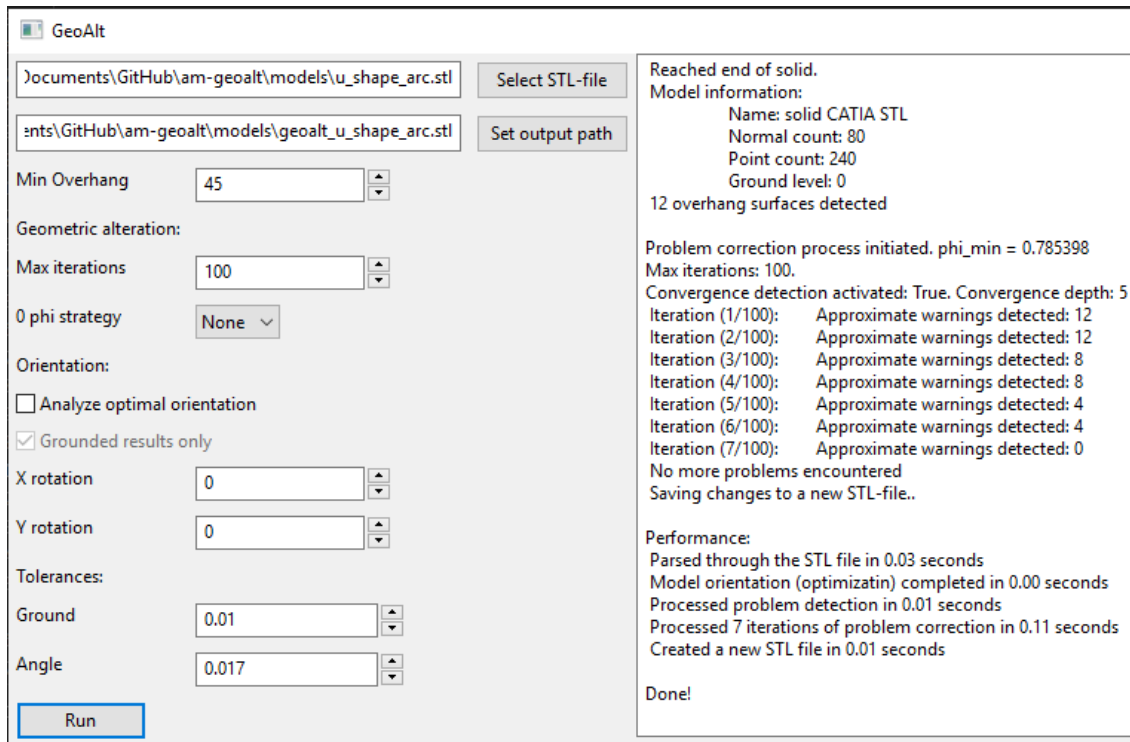
In the end two of these solutions were implemented as optional methods the user can elect to utilize in the final prototype, depending on the situation. Specifically a method for introducing an angle, referred to as "angle injection", and a method for orienting the model to avoid flat overhangs, referred to as "orientation optimization". The reason they were not integrated to be used at all times is because of the varying results these methods would generate. In some cases they would work as intended, while in other cases they broke the models. Any user who would want to attempt to use these implementations would need to be prepared to experiment to get decent results.

## **5.5 Final prototype**

In this section the final version of the prototype will be presented. The final prototype contains both the detection and correction algorithms, as well as an option to optimize the orientation of the model to minimize required support, and an option to utilize the angle injection algorithm to deal with flat overhangs. All of these features together with some other optional settings are packaged into a graphical user interface.



### 5.5.1 User interface



**Figure 5.11:** The final version of the prototype.

To make demonstrations and usage easier of the software that was created during this project, a graphical user interface (GUI) was developed. The GUI makes it simple to select an STL-file, and an output path. The user can then elect to use the correction algorithm (referred to in the GUI as "Geometric alteration", the orientation optimization algorithm, or both. A screen capture of the GUI can be seen in figure 5.11. The GUI provides an array of settings to experiment with:

- Set the input STL file, and the name and path of the output file.
- Customize the minimum allowed overhang, which defaults to 45°
- Determine how long the algorithm is allowed to run before self-terminating by setting an appropriate number of max iterations.
- Choose a "0 phi strategy", which determines how the meta algorithm deals with flat overhangs. In the prototype it is only possible to choose between "None", which ignores the flat overhangs, and "Inject", which attempts to introduce an angle to any encountered flat overhangs. This was briefly touched upon in section 5.4.3.
- Activate or deactivate orientation optimization, which can be run with or without grounded results only.
- If orientation optimization is deactivated, the user can specify a predetermined orientation, to which the model will orient itself immediately.
- Finally, it is possible to set the ground and angle tolerances, which has been fine tuned to 0.01 and 0.017 respectively during development. However, it is

possible that some scenarios calls for these tolerances to be changed.

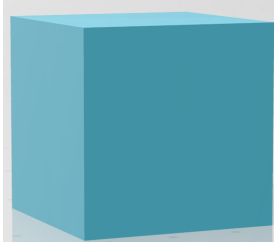


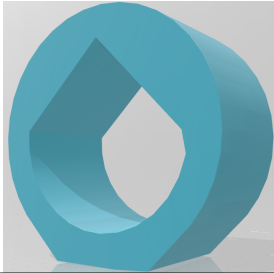
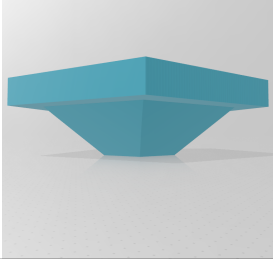
Besides the settings the user can also view the process of the algorithm in the progress output area to the right. This area provides a short description of what the algorithm is working on at any given moment. The program contains a custom made STL file loader which can handle all three STL variations (see 3.2.1), but it can only output ASCII STL-files, which may result in the output file being a lot larger than the input file.

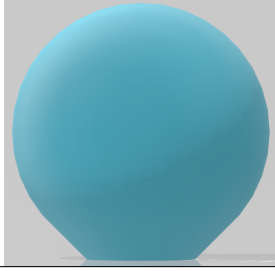
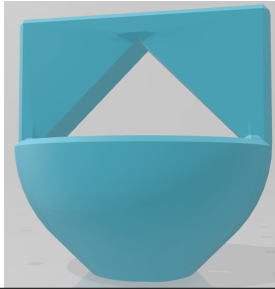
The performance of the final prototype can be seen in section 5.5.2. All benchmark models were run through the final prototype with a minimal overhang set to  $45^\circ$ , and max iterations set to 100. The 0 phi strategy and orientation optimization options were turned off. The tolerance values were left at their default values. All models were successfully treated except for "U-shape 0" which had a flat overhang that could not be treated by the final prototype without the "0 phi strategy" setting set to "inject" (the result of doing this can be seen in section 5.5.4).

### 5.5.2 Performance of the single face algorithm

This section will present the benchmarks as how they looked after being processed by the prototype software. Orientation optimization and angle injection was not utilized. All results are presented in table 5.2.

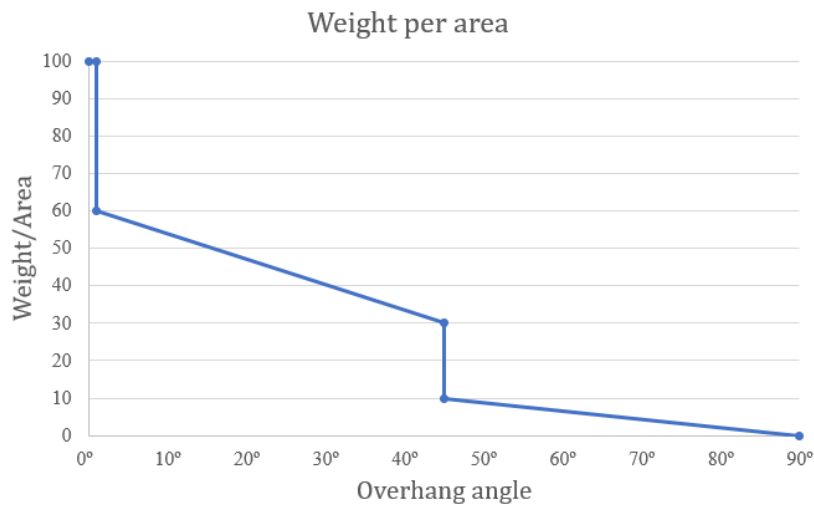
**Table 5.2:** The results of using the final prototype on the benchmark models. In these results orientation optimization and angle injection was turned off. How the models looked before processing can be seen in appendix B.

Label	Model	Notes
Cube		The cube remained unchanged, as expected.
U-shape 0		This model could not be corrected using only the SFA due to the flat overhang. As a consequence this benchmark did not change when run through the SFA algorithm.
U-shape arch		The algorithm correctly pushed the vertices that made up the arch into a triangular shape.
Cylinder		Again, the top half of the inner cylinder has been reshaped into an easier to print triangle. The bottom of the cylinder has also been corrected slightly.
Angular		The algorithm has successfully managed to solve this multi-directional problem. This proves that the change aggregation mechanic works.

Sphere		Again, the sphere requires the change aggregation mechanic to work properly. In this benchmark we can also see the grounded curved surface flattening at the bottom.
Arch-sphere		The algorithm correctly solved what was initially thought to be the hardest benchmark to alter automatically.

As can be seen in the table the final prototype was able to process all of the benchmark models. All problematic overhangs has been eliminated except for the model that is labeled "U-shape 0". This particular model has a flat overhang ( $\varphi_i = 0^\circ$ ), and could for this reason not be treated by the SFA. In sections 5.5.3 and 5.5.4 two methods are depicted that were implemented into the final prototype in an attempt to solve this problem.

### 5.5.3 Orientation optimization



**Figure 5.12:** A diagram of how the weight of a face is determined based on the overhang angle. The value is multiplied by the area of the face, thus giving larger overhang surfaces a larger penalty. In this diagram  $\varphi_{min} = 45^\circ$ . The values in this diagram are not derived from calculations, only trial and error. A more thorough study into orientation optimization would likely yield in a different looking diagram.

In order to address the problem with flat overhangs it was deemed necessary to develop a couple of methods for experimental purposes. The first of these two methods is an orientation optimization algorithm which penalizes solutions that has flat overhangs. This is done by assigning each face a "weight" based on its angle and size, where flat overhangs would receive a very high weight. A diagram of how weight is calculated can be seen in figure 5.12.

The reason the size (or area) of the face is used to calculate the weight instead of the expected support structure volume, is because the purpose of this optimization is to reduce support structure contact, not support structure materials. It is also not always obvious how to calculate the expected support volume, since the support might not always stem from the ground, but instead from another part of the model.

The weight of each face is calculated for every rotation. The total weight is then saved along with the orientation, making it easy to extract the best orientation by looking at which orientation had the lowest total weight.

Not only could it help mitigate the issues related to flat overhangs, but implementing such an algorithm could also be the first step in solving two of the problems which surfaced during the interviews: 1) Existing orientation optimization tools often suggest orientations which require support structures that are hard to remove (for example support structures on rounded surfaces), and 2) Some products have surfaces that should not be in contact with support structures.

If each orientation is to have a weight based on the angle of each face, then it is also feasible to implement a method which also weighs an orientation based on whether or not the product is grounded (meaning, it has at least one flat surface that runs parallel to the ground), and a way of giving a large penalty for any orientation that would require alteration or support of a "strict surface" (a surface that is not allowed to be changed or supported).

An example of how this can be done was developed and implemented into the existing software. It rotates the model in  $5^\circ$  intervals around the X and Y axis (this could easily be changed to a smaller step size, but was intentionally left at  $5^\circ$  per step to save time and resources), leaving the Z axis alone as rotating around the Z-axis does not result in a meaningful change. The rotation is done by multiplying each vertex in the model with a rotation matrix. For each new orientation, the total weight is calculated and stored. When all different orientations has been evaluated the orientation of the model is set to the one which received the lowest weight. To make all options available to the user the 10 best orientations are presented, along with their weight and whether or not they are properly grounded. The user can also elect to only show results that are grounded. The program provides this information to the user through text, an example of which can be seen in figure 5.13.

The orientation optimization algorithm did not work well together with the SFA. The fault lies within the SFA, and how it selects a direction in which to push vertices. As will be discussed further in 6.2.2 it might be possible to solve this problem if given more time. However, it did manage to reduce flat overhangs where possible, and

## 5. Results

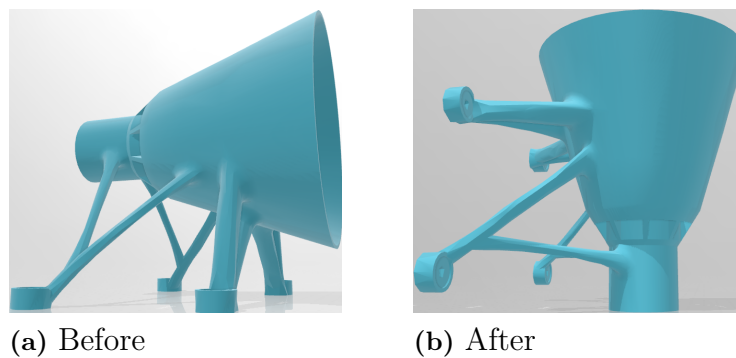
also suggest grounded orientations. The results of the development of the orientation optimization algorithm can be seen in section 5.5.3.

Optimal orientation:	Xrot = 90.00,	Yrot = 0.00,	Weight = 59065.50, Grounded: yes
Alternative 2:	Xrot = 45.00,	Yrot = 140.00,	Weight = 86975.50, Grounded: no
Alternative 3:	Xrot = 135.00,	Yrot = 40.00,	Weight = 86976.26, Grounded: no
Alternative 4:	Xrot = 135.00,	Yrot = 140.00,	Weight = 87107.37, Grounded: no
Alternative 5:	Xrot = 45.00,	Yrot = 40.00,	Weight = 87114.42, Grounded: no
Alternative 6:	Xrot = 130.00,	Yrot = 145.00,	Weight = 87462.73, Grounded: no
Alternative 7:	Xrot = 50.00,	Yrot = 35.00,	Weight = 87471.22, Grounded: no
Alternative 8:	Xrot = 50.00,	Yrot = 145.00,	Weight = 87480.93, Grounded: no
Alternative 9:	Xrot = 130.00,	Yrot = 35.00,	Weight = 87487.24, Grounded: no
Alternative 10:	Xrot = 130.00,	Yrot = 150.00,	Weight = 87535.16, Grounded: no

169 overhang surfaces detected

**Figure 5.13:** Screen capture of the output from the orientation optimization algorithm. The top result has the lowest weight, and is in this case also grounded.

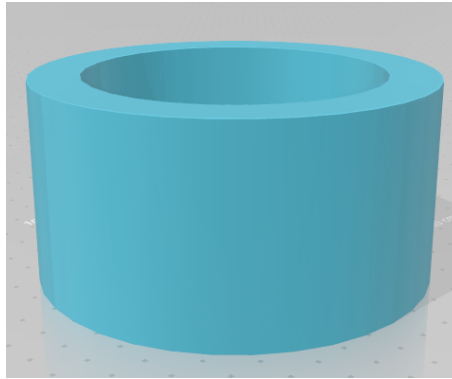
In figure 5.14 a figure of an antenna is shown before and after orientation optimization. In this example the orientation optimization algorithm has managed to find a grounded orientation which should be easier to print than the original orientation. The orientation optimization algorithm has deemed this orientation to be stable because it has a large flat surface touching the ground. It has also attempted to minimize the amount of surfaces that would require support structures. In this situation the bracket structure would need plenty of support, the funnel would likely not require as much support, and none of the support structures would need to be inside of the funnel.



**Figure 5.14:** Orientation optimization of an antenna component. The result is grounded at the base of the antenna, but the bracket structure will need plenty of support if this product is to be printed in the orientation shown in (b). However, in (a) it would be necessary to use support structures inside and underneath the funnel, which would likely be more problematic.

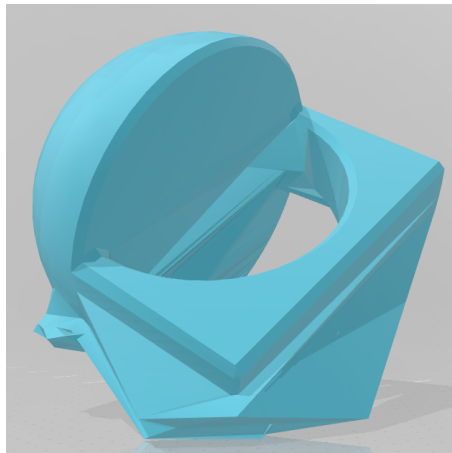
In figure 5.15 the orientation optimization has been used on the cylinder benchmark model. Again, a grounded orientation was produced, with no problematic overhangs. The result in this case is arguably an improved orientation in comparison to the

original, which had plenty of problematic overhangs on the top of the inside of the cylinder.



**Figure 5.15:** The cylinder after being processed by the orientation optimization algorithm. The result is grounded with no problematic overhangs.

In the final figure 5.16 the arch-sphere benchmark model was put through both the orientation optimization algorithm, and the problem correction software. This did not work as intended, as the resulting model has had some of its surfaces distorted to a high degree.



**Figure 5.16:** The arch-sphere benchmark model after run through both the orientation optimization algorithm, and the geometry alteration algorithm. This did not work as intended.

#### 5.5.4 Angle injection

In order to provide an alternative to orientation optimization, a method for introducing angles to flat overhangs was introduced. As mentioned in section 5.4.3, one of the main issues with introducing an angle to a flat overhang is determining in which direction that angle should be introduced. The angle injection algorithm utilizes neighbouring faces to the flat overhang to determine the direction. It looks

for adjacent faces that are on a lower position along the Z-axis. If such a face is found, then the edge that connects the neighbouring face to the targeted overhang is lowered to a midpoint between the lowest and highest vertices of both faces.

During tests this turned out to be an unreliable way to create angles where there originally was none. It worked on simple geometries such as the benchmark model "U-shape 0". However, the algorithm would require much more work in order to be able to successfully process more complicated geometries. Figure 5.17 shows the performance of the angle injection module "U-shape 0". After an angle has been "injected" the single face algorithm processes the new angle. In this way the angle injection algorithm works in concept with the SFA to attempt to solve the problem.



**Figure 5.17:** From left to right: The first image shows the unaltered original model. The second shows the same model, but the flat overhang has had an angle injected. The third shows how the single face algorithm now can take an advantage of the introduced angle, and thus pushes it to form a shape that does not need support.



# 6

## Discussion

### 6.1 Geometry alteration and the space industry

Adding material to a product in order to reduce its need for support structures might reduce the amount of material needed to create it, and it might also reduce time spent in post processing. However, when launching a product into space each gram costs a large amount of money, and so the manufacturer needs to consider the available options carefully. If the cost of materials and post processing of the part is less then the increased cost of launching the altered part, then one of the major benefits of AM (as explained in 1.2) is cancelled out. A solution to this problem could be to use a subtractive geometry alteration algorithm, which removes material rather than adding.

One important thing to note is that, no matter if an additive or subtractive geometry alteration algorithm is used, there will always be a problem with maintaining functionality. Part geometry is often strictly specified by customer requirements. Such components may not be suited for automatic geometry alteration. However, structural components may be better suited for automatic alteration, but would require thorough examination afterwards due to potential changes to mass and mechanical properties.

### 6.2 Further development

If more time were to be put into this project, then there are a number of things that could be explored and improved. This ranges from basic things such as performance, to adding new features.

#### 6.2.1 Improving the performance

The current algorithm has many performance issues that could be resolved. The two most important issues exists within the problem correction algorithm, and in the problem detection algorithm. The problem correction algorithm is single threaded, which means it does not utilize the resources available efficiently. Reworking this could make the algorithm a lot faster. The problem detection algorithm has the same problem. The inefficiency of the problem detection algorithm causes significant slowdown in many parts of the program, as it is used repeatedly inside the meta algorithm, and the orientation optimization. Thus, improving the efficiency of the

problem detection algorithm would likely yield in a large performance boost and should be prioritized if the project is to continue.

### 6.2.2 Improving the results

The two largest issues that was encountered during the project was the flat overhang issue, and the blobbing issue (see 5.3.6 for more information about these issues). These issues prevented the quality of the results from reaching its full potential. While these two problems exists it is hard to motivate using the software for any industrial application. Therefore any further development should focus on solving the issue of flat overhangs and blobbing.

In order to solve the issue of blobbing, the first piece of code that should be put under review is the core of the SFA. More specifically, the choice of pushing direction. It is likely that always pushing vertices in the direction of the normal vector's projection onto the XY-plane is the reason for blobbing, and perhaps there is a better way to extract the preferred direction of movement. Finding a better way to extract a proper vertex pushing direction might also improve the results of using the orientation optimization algorithm together with the SFA.

Another aspect which might help make the results more meaningful would be to take heat dispersion into consideration. Even if all angles are perfect, then situations may arise where, for example, the model is suggested to be printed in a way which makes a thin cross section be printed before a thick cross section. An example of this was presented in the background section of this thesis (see 1.2). A means of compensating for this could be implemented into the orientation optimization, which could analyze the area of vertical slices of the model, and aggregate the results of that analysis into the weight of that orientation.

### 6.2.3 Improving the experience

A feature which would have improved the GUI significantly is a 3D graphical view of the STL-model. This graphical display could be used to visualize different orientations, making it easier for the user to select an optimal orientation after running the orientation optimization algorithm. Such a display could also be used as an interface which the user could utilize to select "strict surfaces" (surfaces that may not be altered, or touched by support structures), which would allow the user an extra layer of control. Implementing support for strict surfaces into the orientation optimization algorithm and the meta algorithm is an easy step, compared to implementing a means for the user to select such surfaces.

### 6.2.4 Improving integration

An idea which was far outside of the scope for this project is to directly apply geometry alteration to a CAD-file. While this is a much more complicated task, it may assist in overcoming some of the obstacles that were encountered when altering the STL format. Primarily, a CAD file can contain constraints and measurements,

which could allow the user to specify to what degree the model is allowed to be modified. By applying such a principle it would be possible to avoid problems with collisions, which was one of the problems which could not be solved during this project. Having the modifications done directly to the CAD model would also make it easier for the designer to control and edit the results.

### **6.3 Use cases**

In its current state the software should not be relied upon to make intelligent decisions regarding a products geometry. However, it may be used as a guide to identify problems, and sometimes it might even help the designer by hinting at a possible solution. But the software does struggle with more complex designs, and might only be of use in "blocky" low complexity models.

If the project is continued, and the software improved upon, then the potential use cases might expand. But even if the software is developed to its full potential it should always be used under supervision. The models needs to be examined after processing to ensure that the functionality is maintained, and any hypothetical specifications are met.

# 7

## Summary

The original question that laid the foundation for this thesis was "How can the geometry of a product be automatically altered to reduce the need for support structures while maintaining product functionality?" (see chapter 2). The results produced during this thesis project shows one way to tackle this problem, but there is still much that needs to be done. The performance of the final prototype on the benchmark models was better than expected, but the benchmark models all had one thing in common: they lacked an actual function. The benchmark models were simply models made to test the software's performance in different types of situations. From that perspective the tests were successful, but they failed to test the software's ability to preserve product functionality.

The software was applied on other models as well, but it quickly ran into problems. Geometries with complicated geometries resulted in "blobbing", which caused lumps to form on some curved surfaces. The most severe problem which was encountered was that of "flat overhangs", in other words overhang surfaces that point straight down at the substrate. Such surfaces completely lack an overhang angle, and could thus not be treated by the algorithm which requires at least a small angle in order to know in which direction to alter the geometry. Two separate algorithm was created which attempted to deal with flat overhangs. These algorithms were added as modules to the final prototype. The first module changed the orientation of the model to reduce the changes of flat overhangs. The other module directly treated flat overhangs, but could only handle very basic flat overhang cases. These modules proved that the problem is not impossible to solve, but requires a lot more work. As for the lumps that were erroneously formed on complex curvature, this issue is likely to be a bug in the SFA (5.3.3), and could probably be resolved if given enough time.

The final question is, could the space industry benefit from this sort of technology? During the interview with the manager from within the space industry it was revealed that customer requirements sets yields in strict specifications on geometry. And so it is often the case that geometry can only be altered within a narrow tolerance before becoming unusable. This does not necessarily mean that the technology is useless, as it could still be used to give a designer helpful feedback on the manufacturability of their model, and guide them in the right direction.

Adding additional features to the technology could result in it being useful not only for feedback, but also for finalizing a design. One of these features could be the ability of a designer to select certain surfaces that are not allowed to be altered.

This would result in the software skipping those surfaces, which would help preserve functionality. Another step towards taking this technology to the next level would be to have it operate directly on CAD models. Since CAD models can contain a lot more information than an STL-file, including measurements, tolerances and many other attributes, the software could incorporate this information into its decision making to yield better results.

In conclusion, this project resulted in the development of a proof of concept in the form of a prototype. It is a long way from a final fully functional product. Geometry can be altered automatically to reduce the need for support structures, but a separate mechanism needs to be developed which preserves product functionality, such as the ability of the user to specify alteration constraints.



# Bibliography

- [1] Ian Gibson, David Rosen, and Brent Stucker. *Additive Manufacturing Technologies. [electronic resource] : 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. New York, NY : Springer New York : Imprint: Springer, 2015., 2015.
- [2] Dimitar Dimitrov, Kristiaan Schreve, and Neal de Beer. Advances in three dimensional printing—state of the art and future perspectives. *Rapid Prototyping Journal*, 12(3):136–147, 2006.
- [3] Christo Dordlofva, Angelica Lindwall, and Peter Törlind. Opportunities and challenges for additive manufacturing in space applications. In *12th Biennial Norddesign 2016 Conference "Highlighting the Nordic Approach", Trondheim, Norway, 10-12 August 2016*, volume 1, pages 401–410. The Design Society, 2016.
- [4] William E Frazier. Metal additive manufacturing: a review. *Journal of Materials Engineering and Performance*, 23(6):1917–1928, 2014.
- [5] M Cloots, A Spierings, and K Wegener. Assessing new support minimizing strategies for the additive manufacturing technology slm. In *Solid Freeform Fabrication Symposium (SFF), Austin, TX, Aug*, pages 12–14, 2013.
- [6] Daniel Thomas. *The development of design rules for selective laser melting*. PhD thesis, University of Wales, 2009.
- [7] Steven C Wheelwright and Kim B Clark. *Revolutionizing product development: quantum leaps in speed, efficiency, and quality*. Simon and Schuster, 1992.
- [8] Angelica Lindwall, Christo Dordlofva, and Anna Öhrwall Rönnbäck. Additive manufacturing and the product development process: insights from the space industry. In *21st International Conference on Engineering Design (ICED17), Vancouver, Canada, 21-25 August 2017*, volume 5, pages 345–354, 2017.
- [9] Christian Lindemann, Ulrich Jahnke, Matthias Moi, and Rainer Koch. Analyzing product lifecycle costs for a better understanding of cost drivers in additive manufacturing. In *23th Annual International Solid Freeform Fabrication Symposium—An Additive Manufacturing Conference. Austin Texas USA 6th-8th August*, 2012.
- [10] Jeff Allen. An investigation into the comparative costs of additive manufacture vs. machine from solid for aero engine parts. Technical report, ROLLS-ROYCE PLC DERBY (UNITED KINGDOM), 2006.
- [11] Giorgio Strano, L Hao, RM Everson, and KE Evans. A new approach to the design and optimisation of support structures in additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 66(9-12):1247–1254, 2013.

- [12] Materialise magics support structure generation software. <https://www.materialise.com/en/software/magics/modules/metal-support-generation-module>. Accessed: 2019-06-14.
- [13] LongFei Qie, ShiKai Jing, RuiChao Lian, Ying Chen, and JiHong Liu. Quantitative suggestions for build orientation selection. *The International Journal of Advanced Manufacturing Technology*, 98(5-8):1831–1845, 2018.
- [14] Martin Leary, Luigi Merli, Federico Torti, Maciej Mazur, and Milan Brandt. Optimal topology for additive manufacture: A method for enabling additive manufacture of support-free optimal structures. *Materials & Design*, 63:678–690, 2014.
- [15] Space junk could destroy satellites, hurt economies. <https://phys.org/news/2017-05-space-junk-satellites-economies.html>. Accessed: 2019-01-27.
- [16] Space debris and human spacecraft. [https://www.nasa.gov/mission\\_pages/station/news/orbital\\_debris.html](https://www.nasa.gov/mission_pages/station/news/orbital_debris.html). Accessed: 2019-01-27.
- [17] John Mark Mattox. Additive manufacturing and its implications for military ethics. *Journal of Military Ethics*, 12(3):225–234, 2013.
- [18] Samuel H Huang, Peng Liu, Abhiram Mokasdar, and Liang Hou. Additive manufacturing and its societal impact: a literature review. *The International Journal of Advanced Manufacturing Technology*, 67(5-8):1191–1203, 2013.
- [19] Di Wang, Yongqiang Yang, Ziheng Yi, and Xubin Su. Research on the fabricating quality optimization of the overhanging surface in slm process. *The International Journal of Advanced Manufacturing Technology*, 65(9-12):1471–1484, 2013.
- [20] D Brackett, I Ashcroft, and R Hague. Topology optimization for additive manufacturing. In *Proceedings of the solid freeform fabrication symposium, Austin, TX*, volume 1, pages 348–362. S, 2011.
- [21] The European Powder Metallurgy Association (EPMA). Introduction to additive manufacturing technology, 2nd edition, 2017.
- [22] Stl file format (3d printing) – simply explained. <https://all3dp.com/what-is-stl-file-format-extension-3d-printing/#pointone>. Accessed: 2019-02-20.
- [23] Voxelization. <https://labs.cs.sunysb.edu/labs/projects/volume/Papers/Voxel/index.html>. Accessed: 2019-05-12.
- [24] Karl T. Ulrich and Steven D. Eppinger. *Product design and development*. McGraw-Hill/Irwin, 2011.

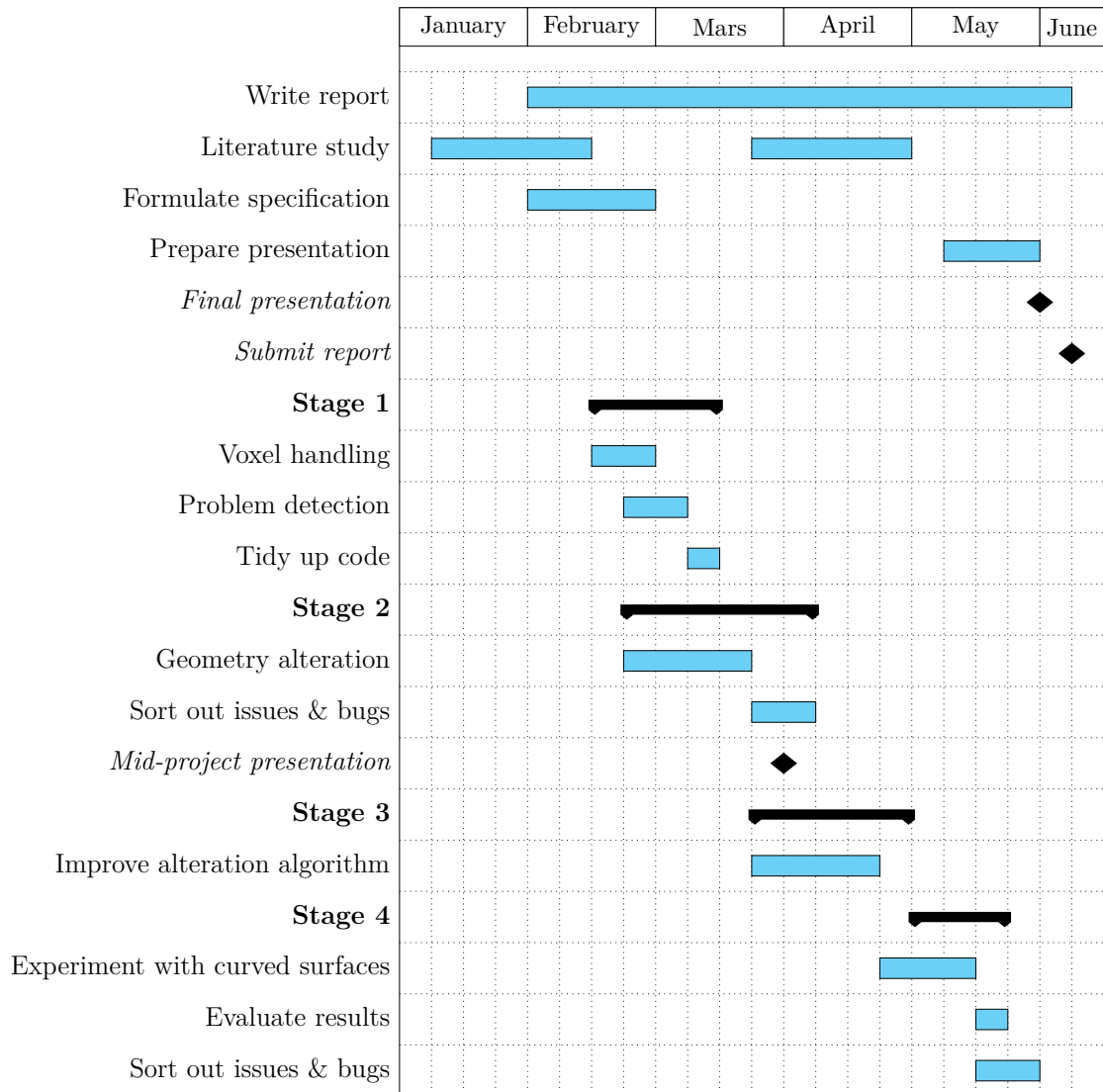


# Appendices

# A

## Timetable

Figure A.1 shows the Gantt chart for the project. This Gantt chart represents the ideal scenario, and is thus prone to change as new information is uncovered. Towards the end of stage 2 a mid-project presentation will be held. Besides the tasks shown in the chart there will also be weekly meetings with the project supervisor.

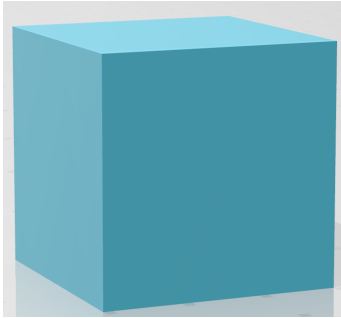
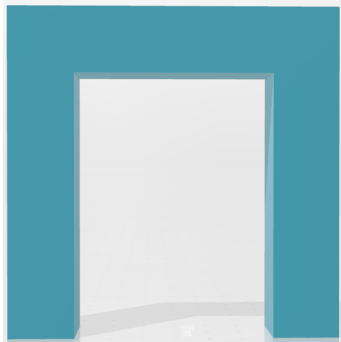



**Figure A.1:** This figure displays the ideal Gantt chart for the project.

# B

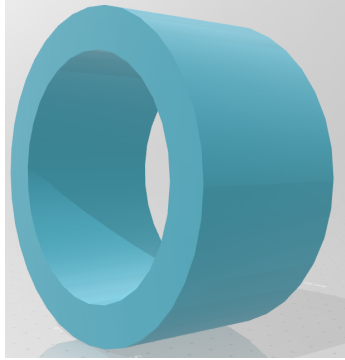
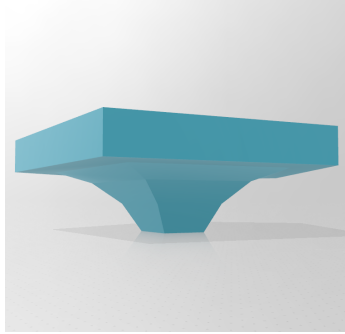
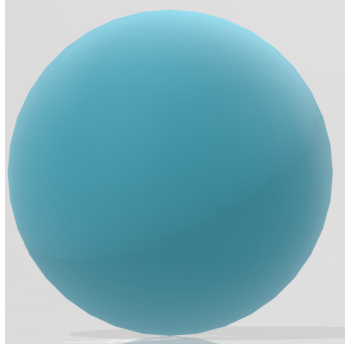

## Benchmarks

In this section the unprocessed benchmarks are presented. Along with their pictures and names, there is also a short explanation of the characteristics that makes each benchmark important.

Label	Model	Characteristics
Cube		Possesses no problems, and should remain untouched by the algorithm.
U-shape 0		Possesses a problematic flat overhang.
U-shape arch		Possesses an overhanging arch structure with several bad angles.

## B. Benchmarks

---

Cylinder		A cylinder lying parallel to its internal axis provides a similar problem to the U-shape arch, but also has an outside surface with problematic faces.
Angular		This is the first benchmark that has problems in more than one plane (all previous benchmark models has had problems only in the XZ-plane)
Sphere		Similar to the "Angular" benchmark, the sphere has problems in all directions, but higher model complexity.
Arch-sphere		A combination of the "Sphere" benchmark and the "U-shape arch" benchmark, ensuring that one single algorithm can handle two vastly different problems in the same process.