NTNU

Norwegian University of
Science and Technology

An Additive Manufacturing Path Generation Method Based on CAD Models for Robot Manipulators

Ingrid Fjordheim Onstein

Master of Science in Cybernetics and Robotics
Submission date: May 2018
Supervisor: Jan Tommy Gravdahl, ITK
Co-supervisor: Linn Danielsen Evjemo, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem description

Main objectives for this thesis:

1. Do a literature review on the state-of-the-art of large-scale additive manufacturing. Consider the benefits and challenges of using a 6 DOF industrial robot as opposed to using a traditional 3 DOF translation-only set-up.

2. Present the methods and software currently used for generating trajectories for extrusion based AM based on CAD models.

3. Improve the method from the preliminary project that used available software to generate a trajectory for printing a simple geometrical form. Evaluate whether or not this improved method is suitable for further work on large-scale AM with a 6 DOF robot.

4. Develop a new method to generate a path for printing simple geometrical forms using robot manipulators that takes advantage of the robots 6 DOF. This includes generating a path that reduce the need for support structures and that can print along curved contours.

Abstract

Traditional extrusion based Additive Manufacturing (AM) is realized using a 3 Degrees of Freedom (DOF), translation only, 3D printer. Here, the printer must be larger than the printed part. One way of enabling AM in large-scale is to combine AM with robotics. By using a 6 DOF robot manipulator to extrude a fast-curing material, the workspace of the build would be greatly expanded and it would be possible to increase the flexibility of the building process itself since the structure would no longer have to be built from the bottom-up approach which is necessary for most existing forms of AM. This could possibly reduce the need for support structures to the point of only relying of anchoring and stabilizing. In this thesis, a method for generating a path for AM using robot manipulators that takes advantages of the robots DOF is presented. The path is generated based on simple surfaces in CAD models. First, the surface(s) is sampled and the samples are gathered in a point cloud. Then, a path is generated based on the point cloud using a path generation algorithm. Three different path generation algorithms was implemented and tested: greedy choice, weighted greedy choice and Travelling Salesman Problem (TSP). Out of the three algorithms, the weighted greedy choice algorithm shows the most promise. With this algorithm, paths that enable printing along curved surfaces and reducing the need for support structures was generated. The method is effective, and by interfacing with FreeCAD, it is easy to review the generated paths through visual aids. It is, however, important to mention that the method only generates paths based on simple surfaces and is based on the assumption of fast-curing material enabling mid-air printing.

Sammendrag

Til tradisjonell 3D-printing brukes 3D-printere med 3 frihetsgrader. Dette medfører at printeren må være større enn delen som skal printes. 3D-printing i storskala kan muliggjøres ved å kombinere 3D-printing og robotikk. Ved å bruke en robotmanipulator med 6 frihetsgrader til å ekstrudere et materiale som herder raskt, kan arbeidsområdet bli kraftig utvidet og det vil være mulig å øke flesibiliteten til byggeprosessen. Dette er siden strukturen ikke lenger må bygges lag for lag fra bunnen av som er nødvendig for de fleste eksisterende løsninger innen 3D-printing. Dette kan redusere behovet for støttestrukturer til kun å være avhengig av forankring og stabilisering.

I denne avhandlingen, vil en metode for å generere baner til 3D-printing for en robotmanipulator bli presentert. Målet med metoden er å utnytte alle de 6 frihetsgradene. De genererte banene er basert på enkle overflater i CAD-modeller. Det første steget i prosessen er å sample de valgte overflatene og samle alle punktene i en felles punktsky. Deretter vil en en bane bli generert basert på denne punktskyen ved hjelp av en algoritme. Tre forskjellige algoritmer er implementert og testet, herunder grådig algoritme, vektet grådig algoritme og travelling salesman problem (TSP). Blant de tre algoritmene er det vektet grådig som viser mest potensiale. Denne algoritmen genererer baner som muliggjør 3D-printing langs kurvede overflater og som reduserer behovet for støttestrukturer. Metoden er effektiv og gjennom FreeCAD er det lett å vurdere de genererte banene ved hjelp av visuelle verktøy. Det er viktig å nevne at metoden kun genererer baner basert på enkle overflater, og er basert på antagelsen om materiale som herder hurtig.

Preface

This thesis is written as part of a Master's degree in Engineering Cybernetics at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). The project was carried out from January 2018 to June 2018. This thesis is based on the work done in the specialization project performed by the student with the title "Additive manufacturing by robot manipulator: A method for generating robot trajectory based on CAD model using existing solutions". The task was formulated as an open problem and the student has chosen method and software independently. Guidance was received from supervisor Jan Tommy Gravdahl, co-supervisor Linn Danielsen Evjemo and from Mathias Hauan Arbo, all at NTNU.

Acknowledgements

I would like to thank my supervisor and co-supervisor at NTNU, Jan Tommy Gravdahl and Linn Danielsen Evjemo, for their help and guidance throughout this project. I would also like to thank Mathias Hauan Arbo at NTNU for discussing the project with me when needed. Finally, I would like to thank my brother, Håkon Fjordheim Onstein, for providing me with CAD models on request.

Contents

List of Figures

Abbreviations

AM Additive Manufacturing

B-rep Boundary representation

CAD Computer Aided Design

CAM Computer Aided Manufacturing

CLI Common Layer Interface

CNC Computer Numerical Control

DOF Degrees of Freedom

FDM Fused Deposition Modeling

ISO International Standards Organization

NURBS Non-uniform rational B-splines

OLP Offline Programming

ROS Robot Operating System

SM Subtractive Manufacturing

STEP Standard for the Exchange of Product Model data

STL Stereo Lithography

TSP Travelling Salesman Problem

WAAM Wire Arc Additive Manufacturing

Chapter 1

Introduction

1.1 Motivation

In the recent years, the development of AM has increased rapidly and AM has become more available to the public, Evans (2012). AM is the formalized term for what used to be called rapid prototyping or what is more popularly called 3D printing, Gibson et al. (2010). It works by adding material parts in layers; each layer a thin cross-section of the part derived from the original Computer Aided Design (CAD) model. Initially AM has been used to create visualization for products as they were being developed, but today, AM is used to fabricate end-use products for aircraft, dental restoration, medical implants, automobiles and even fashion products, *AM basics* (2017). Traditionally, AM is realized using a 3D printer that can move in three linear directions along the x-y-z axes, Evans (2012). It then follows that the 3D printer must be larger than the print. To print in large scale using traditional 3D printers, the part must be printed in smaller sub-parts before being mounted together afterwards. Another way of enabling AM in large scale is to combine AM with robotics, Evjemo et al. (2017). By using a 6 DOF robot manipulator to extrude a fast-curing material, the workspace for the build could be greatly expanded and it would also be possible to increase the flexibility of the building process itself because the structure would no longer have to be built layer by

layer from the bottom-up approach which is necessary for most existing forms of AM. This could reduce or even remove the need for support structures to the point of only relying on anchoring and stabilizing.

In the specialization project, which is presented in chapter 3, it was investigated how a robot manipulator trajectory for extrusion based AM could be generated based on a CAD model using existing solutions. A method that enabled AM using robot manipulators was found, but it did not fully take advantage of the robots DOF. This thesis will further investigate how a path can be generated based on a CAD model for extrusion based AM using a robot manipulator. The goal is that the path takes advantage of the robot manipulators DOF. A method for generating such paths will be developed and tested on different CAD models. The thesis will start with a literature review on the state-of-the-art AM using industrial manipulators, challenges and benefits of using a 6 DOF robot manipulator compared to a traditional 3 DOF 3D printer, a review of the methods and software currently used for AM based on CAD models and a overview of CAD models and how they are built up.

1.2 Literature review

There has been a lot of development within AM in the recent years. Livesu et al. (2017) present an overview of the AM processing pipeline from 3D model to 3D print. Evjemo et al. (2017) present an overview of state-of-the-art AM by robot manipulator including a proof-of-concept experiment. *MX3D* (2018) and *Stratasys* (2017) are both commercial companies that present novel use of AM using robot manipulators, MX3D for large scale printing and Stratasys for removing the need of support structures. Within the subject of using robot manipulators for AM with non-planer contours, Zhang et al. (2015), Zhang et al. (2016) and Alsharhan et al. (2017) show promising results with increased material properties compared to planar contours. There has also been some resent development with using arc-welding techniques for AM, so called Wire Arc Additive Manufacturing (WAAM). This is attracting interest from the manufacturing industry because of their potential to fabricate large metal components with low cost and short production lead time. A review of WAAM is presented in Pan et al. (2018).

1.3 Assumptions

There are two main assumptions that will be adopted throughout this thesis.
Assumption 1. The material is fast-curing enabling printing in mid-air.
Assumption 2. Only simple surfaces of the CAD model is used.
Remark 1. Tool-path planning is out of the scope of this thesis.

1.4 Contributions

The work in this thesis is based on the specialization project presented in chapter 3.

The following contributions have been made in this thesis:

- An overview of both traditional AM and AM using robots including different technologies, methods and existing solutions.
- Further work on the method presented in the preliminary project for generating robot trajectories for AM based on CAD models for simple geometrical forms using existing solutions. The method is tested and verified using the virtual controller in RobotStudio.
- A novel method for generating paths for AM using robot manipulators that takes advantage of the robots DOF. The method consists of three different path generation algorithms: greedy choice, weighted greedy choice and Travelling Salesman Problem (TSP).
- The novel method is implemented using Python and FreeCAD.
- Studies evaluating the performance of the novel method on two different CAD models, one with double curvature and one with overhang.
- Considered future work necessary to take this method to use

1.5 Outline

This thesis is divided into seven chapters. After this introductory chapter, there is a literature study on the existing solutions for both traditional AM and AM using robots, Computer-Aided Design and its standards, the state-of-the-art AM methods and software, robot manipulator control including available software and finally a section on different algorithms for path generation. This thesis is based on the work from the specialization project which is presented in chapter 3. Chapter 3 also includes some further work and results including a discussion of why a new and improved method was necessary. In chapter 4, the new method for generating paths for AM is presented. The result from this method is presented in chapter 5. The results are discussed in chapter 6 before concluding and considering future work in chapter 7.

Chapter 2

Literature study

This chapter is based on the chapter with the same title in the preliminary project, except for section 2.2 about CAD and section 2.5 about algorithms.

2.1 Additive manufacturing

AM is the formalized term for what used to be called rapid prototyping and what is popularly called 3D printing, Gibson et al. (2010). AM works by adding material parts in layers; each layer is a thin cross-section of the part derived from the original CAD model. The counterpart of AM is Subtractive Manufacturing (SM) such as machining which is defined by *Machining* (2017) as a process where a piece of raw material is cut into a desired final shape and size by a controlled material-removal process. Reasons for using AM instead of SM can be that SM is not able to produce highly complex parts, AM can be much faster and that AM can be much cheaper. Initially, AM was used to create visualization for products as they were being developed. As the AM technology has developed, the number of applications has increased. Today, AM is used to fabricate end-use products in aircraft, dental restorations, medical implants, automobiles, and even fashion products, *AM basics* (2017).

2.1.1 Using 3D printers

To print a 3D model, a 3D printer is used. A traditional 3D printer can be compared to a 3 DOF Cartesian robot. This is a machine that can move in three linear directions along the x-y-z axes, Evans (2012). 3D printers have been around for engineers the last 25 years, but become available to the public only the last few years. There are two main AM technologies for layer by layer 3D printing; local deposition of material and solidification of material, Livesu et al. (2017).

Material deposition refers to methods that create the next layer by locally depositing material on a previously printed layer. One example of a material deposition method is Fused Deposition Modeling (FDM). FDM is defined as systems that builds parts layer-by-layer by depositing semisolid molten polymeric materials in the shape of thin filament (or road/bead) via computer-controlled robotic extruder, Taufik and Jain (2016). There are several advantages of material deposition such as the ability to combine multiple materials, printing time that mostly depend on the part volume and the ability to fully enclose voids. A huge drawback however, is the strong requirement for support structures, Livesu et al. (2017).

Layer solidification refers to all the processes that build the object by solidifying the top (or bottom) surface of a non-solid material, typically within a tank. Examples of layer solidification methods are photo-polymerization, powder bed fusion and sheet lamination, Livesu et al. (2017). The main advantage of layer solidification compared to material deposition is that the need for support structures are reduced. The major drawback however, is that layer solidification typically require a tank larger than the print with non-solid material. This makes this method very impractical for large-scale AM.

2.1.2 Using industrial robots

AM can be time demanding for larger components since the layer size affect the roughness and accuracy of the constructed surface. As a result, AM-machines are often restricted to small build volumes. However, for parts where surface quality and detail is less important, it is possible to build quicker and bigger. For traditional AM-processes,

this implies that the machines need to grow to a larger scale than the produced parts. One way of enabling AM in large scale is to combine AM with robotics. By using a 6 DOF robot manipulator to extrude a fast-curing material, the workspace for the build could be massively expanded and it will also be possible to increase the flexibility of the building process itself because the structure would no longer have to be built layer by layer from the bottom-up which is necessary for most existing forms of AM. This could reduce or even remove the need for support structures to the point of only relying on anchoring and stabilizing. One possibility is also to make several robot manipulators work simultaneously, possibly with different materials, on the same part, Evjemo et al. (2017).

2.1.3 Existing solutions

3D printed houses

Over the last decade, there has been great progress in the field of 3D printing houses. Two existing projects are The BOD and New Story + ICONs 3D printed houses. Both are realized using a large scale 3D printer.

In Denmark, the construction of Europe's first 3D printed building started in 2017, *The construction of Europe's first 3D printed building has begun* (2017). The goal of the project is to demonstrate how 3D printing technology can be applied for constructions in Europe, still fulfilling the usual European building codes. The BOD, which is short for Building on Demand", will be a 3D printed small office hotel of less than 50 square meters located in Copenhagen. It does not contain any straight lines of walls except for the windows and doors, and is printed using the bottom-up approach. A picture of the planned result can be seen in figure 2.1. The printing of the BOD is completed, but windows, doors and roof are still to be mounted.

The other 3D printed house is made by Icon and New Story. In March 2018, they showed their first 3D printed house in Austin, Texas. The house is 32 m^2 and is printed by a printer, made up of aluminum, in concrete layer by layer, *New Story + ICON: 3D printed houses* (2018). It is stated that it will take less than 24 hours to print a house and that the price of one house will be $4.000. The goal of the project is to bring homes

Figure 2.1: The BOD. Image courtesy of *The construction of Europe's first 3D printed building has begun* (2017)

Figure 2.2: New Story + ICON: 3D printed house. Image courtesy of *New Story + ICON: 3D printed houses* (2018)

to the underdeveloped parts of the world. In figure 2.2 there is a picture of the printed house in Austin, Texas.

Using industrial robots

In the Netherlands, Joris Laarman Lab enables craftsmen, scientists and engineers to collaborate on emerging technologies such as 3D printing, *Joris Laarman Lab* (2018). In 2012, Joris Laarman Lab, together with the Institute pf Advanced Architecture of Catalonia (IAAC), started the Mataerial project, *Mataerial* (2018) The project used a 6 DOF industrial manipulator to deposit material along a pre-designed trajectory. This resulted in a patented AM method for building fast-curing thermoplastic in any direction. There is a picture showing the result of the Mataerial project in figure 2.3. This picture shows that by combining industrial manipulators with a fast-curing plastic

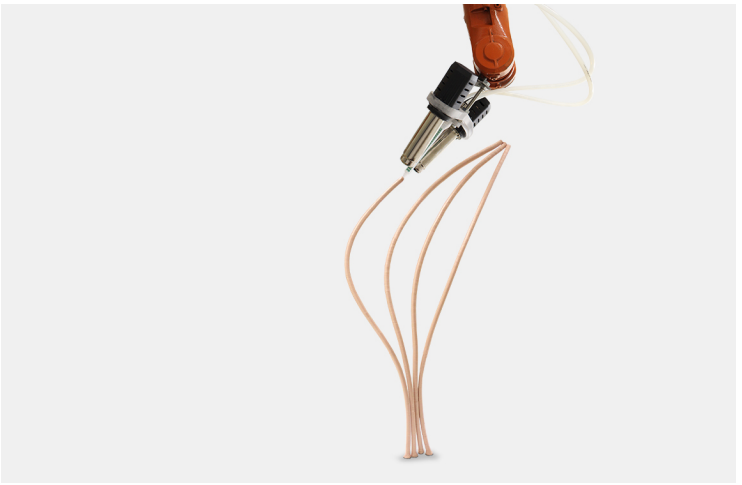


Figure 2.3: The Mataerial project enables AM along double curved lines in mid-air. Image courtesy of *Mataerial* (2018)

makes it possible to "print" along double curved lines in mid-air.

Another project to come out of the Joris Laarman Lab is MX3D, *MX3D* (2018), and this is one of the projects that has come the furthest within AM using robot manipulators. They have successfully printed a butterfly screen and a gradient screen in metal using a robot manipulator. Their newest project is to print a fully functional stainless steel bridge that is to cross one of the canals in the center of Amsterdam. The bridge is printed in parts of roughly one meter which is then assembled together afterwards. In September 2017, they managed to mount the robot manipulator directly on the bridge to print horizontally. A picture of the project from September 2017 is shown in figure 2.4. April 2018, MX3D completed the full span of the bridge, only missing the swirls on the end of the bridge. A picture of the complete span of the bridge is shown in figure 2.5.

Another company that has come quite far within AM using robots is Stratasys, *Stratasys* (2017). Stratasys has developed a new system called Robot Composite 3D Demonstrator, *Stratasys demonstrates next generation 3D printing technology for large*

Figure 2.4: MX3D Bridge Project - Robot on bridge. Image courtesy of *MX3D* (2018).

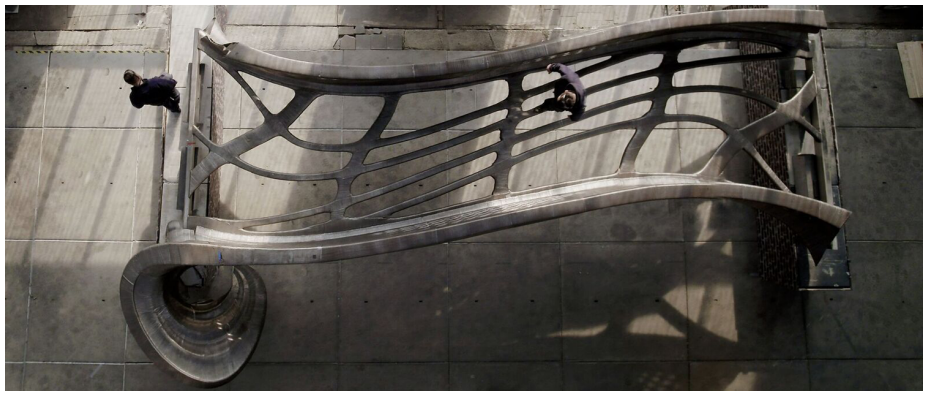


Figure 2.5: MX3D Bridge Project - Complete span. Image courtesy of *MX3D* (2018)

Figure 2.6: Part printed by the Robot Composite 3D Demonstrator. Image courtesy of *Robotic composite 3D demonstrator* (2017).

parts composites (2016). In the demonstrator, they have combined their own AM processes with Siemens's industrial motion control hardware. The result is an 8 DOF system that allows for precise directional material placement while reducing the need for support structures. The demonstrator is put together by one robot manipulator with 6 DOF and one smaller robot with 2 DOF. The two robots work together allowing the demonstrator to print along any X-Y-Z path. In figure 2.6 is a picture of the printed part.

In Zheng et al. (2017), a method for automatic CAD-based path generation and optimization for laser cladding robot in AM is presented. The process convert a CAD model into a Stereo Lithography (STL) model, slices the STL model into a point cloud, order the intersections in the x-y plane of the point cloud before finally generating a layer by layer robot path. An experiment using the new method has also been performed which shows promise.

Non-planar contours

Traditionally, all slicing algorithms use planar slices. There has, however, also been a lot of research on the subject of using robot manipulators for printing along non-planar surfaces. Zhang et al. (2015) present a method to fabricate parts or structures by printing along curved surfaces with use of industrial robots. To print, an extrusion head was mounted on the tip of an IRB140 robot. The curved path machine instruction was generated using Arevo Kepler engine software and converted into RAPID code using RobotStudio. Zhang et al. (2016) build on the work of Zhang et al. (2015) and presents a method for robotic AM process simulation which pushes one step further in the direction of design and analysis of AM part with building parameters in consideration. The simulation is realized using RobotStudio and its tools such as tool trace. Also within the work on non-planar surfaces has Alsharhan et al. (2017) developed a method for enhancing mechanical properties of thin-walled structures with non-planar extrusion based AM. The tool path is generated in Matlab and parsed into RAPID before simulated and realized using RobotStudio and an ABB IRB120 robot manipulator. Results from a test that compare the mechanical properties of the same model printed using planar and curved contours are presented and discussed. The results shows that the material properties of the part realized using curved contours are greatly improved compared to using planar contours.

Wire Arc Additive Manufacturing

Arc-welding based AM techniques are attracting interest from the manufacturing industry because of their potential to fabricate large metal components with low cost and short production lead time, Pan et al. (2018). In WAAM, welding equipment is mounted on the tip of a robot manipulator. WAAM has received less attention than other AM processes through time due to high heat input, poor accuracy, no smooth surfaces, immature CAD-to-part AM system and lack of integrated, reliable process monitoring. However, WAAM has been widely explored over the last three decades. This has led to several strategies within both slicing and path planning. This will be explained further in section 2.3.2 and 2.3.3.

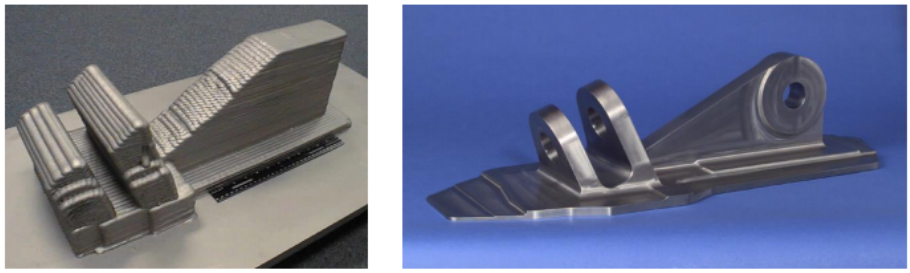


Figure 2.7: A typical aerospace component formed by electron beam wire deposition followed by finishing material removal processes. Image courtesy of Hoye (2015)

In figure 2.7, an aerospace component before and after machining is shown. The first picture shows the the component after AM and the second picture shows the same component after undergoing machining. Machining is necessary for most components made using WAAM.

Surface finishing

The opposite of AM is SM, also known as machining. There are a few areas within machining that is highly applicable within AM as well. One is free form surface finishing. Surface finishing is a broad range of industrial processes that alters the surface of a work piece to achieve a certain property, *Metal Finishing Industry* (1995). Free form surfaces, also called sculptured surfaces, can be defined as surfaces containing one or more non-planar, non-quadratic surfaces generally represented by parametric and/or tessellated models, Lasemi et al. (2010).

There are three major types of tool path topologies: direction-parallel, contour-parallel and space-filling curve (SFC). In Lin et al. (2015), a new method is presented that formulates the problem as a TSP. Here, the optimal path is generated in two steps. The first step is to generate a set of regular cutter contact points on the surface and the second step is to feed the generated points into a TSP solver for generation of the optimal point linking sequence. A figure showing the result of the different tool path generation algorithms is shown in figure 2.8.

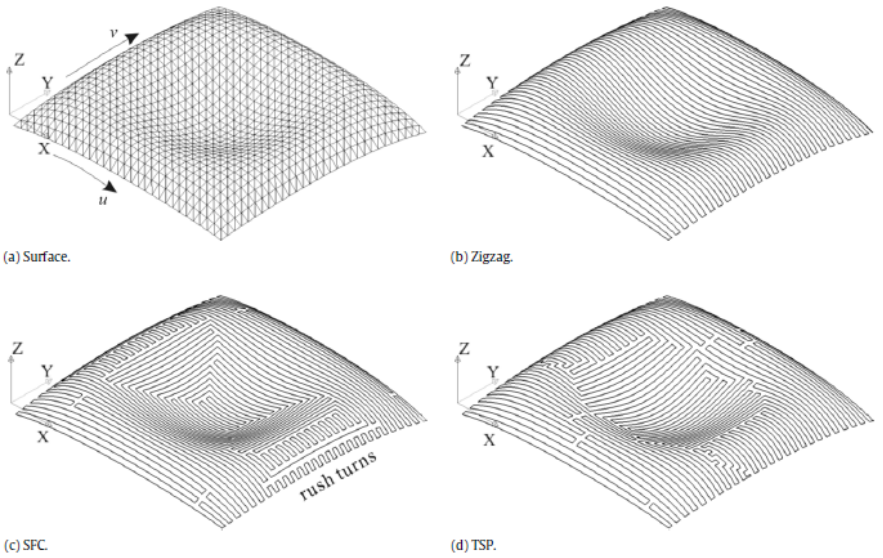


Figure 2.8: Path comparison on a bowl-like surface: (a) the surface; (b) zigzag path; (c) ISFC path; (d) TSP path. Image courtesy of Lin et al. (2015)

2.2 Computer-Aided Design

CAD can be defined as the use of computer systems to assist in the creation, modification, analysis or optimization of a design, Sarcar et al. (2008). It can be used to increase the productivity of the designer, to improve the quality of design and to improve communications through documentation. A CAD model is typically represented using boundary representation.

2.2.1 Boundary representation

Boundary representation (B-rep) represent objects in terms of the "skin" surrounding them, Stroud (2006). The skin is composed of a set of adjacent bounded surface elements, called faces. Faces are bounded by sets of edges, which are portions of curves lying on the surface of the faces on either side of the edge. The points where several faces meet are called vertices. A model showing how B-rep models are built up is shown in figure 2.9. Each object can be divided into two basic groups: topology and geometry. The topology is responsible for defining the structure of the object while the geometry define the form or shape of the object. Examples of topology elements are face, edge and vertex and examples of geometry elements are surface, curve and point.

Free-form geometry

Geometry can be split into two categories: analytical and numerical, Stroud (2006). With analytical geometry, the shape information is explicit. Examples of this is that a curve is circular or that a surface is planar. Analytical geometry can be represented as

$$y = f(x) \quad f(x, y) = 0 \quad f(x, y, z) = 0 \tag{2.1}$$

For numeric geometry, there is no inherent notion of the shape that is represented. The shape is controlled by the positions of a set of points, called the control points, with different weighting functions. The weighting functions are called "basis functions" and different sets of functions give different geometry. Examples of basis functions are

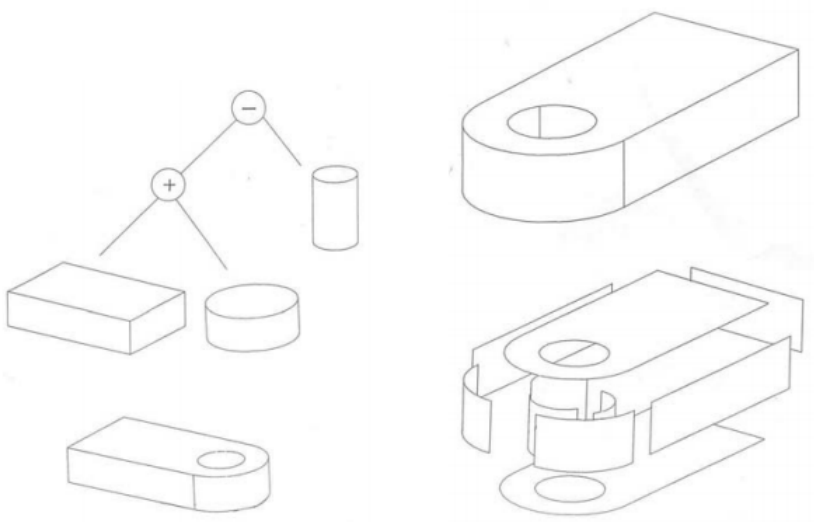


Figure 2.9: B-rep models. Image courtesy of Stroud (2006)

the Bézier form, Basis spline (B-spline) and Non-uniform rational B-splines (NURBS). Numeric geometry is more difficult to use, but useful for representing surfaces where no analytic representation exist. Examples of such surfaces are automobile bodies, ship hulls, sculptures, bottles, shoes, etc, Rogers (2000).

The following descriptions of the different basis functions are taken from Stroud (2006) and Rogers (2000).

Bézier curves was developed by P. Bézier at Renault Automobile Company. It is defined by a set of control points B_0 through B_n where n is called its order. The curve passes through the first and terminating point while the remaining points are used to define the slope of the curve at the end points.

The general formula for a Bézier curve of degree n is

$$P(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-1)!}(1-t)^{n-i}t^i B_i \tag{2.2}$$

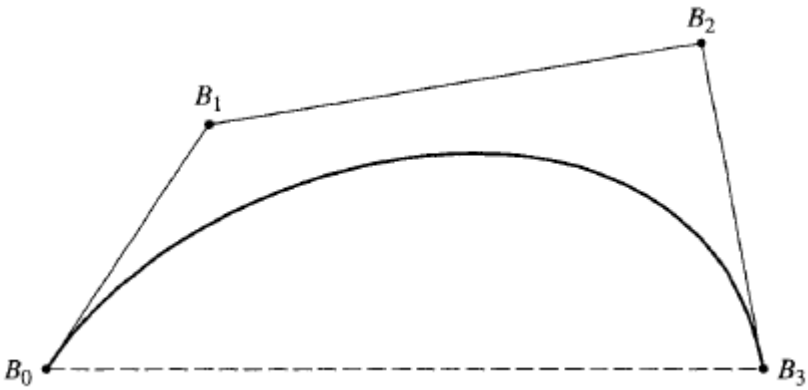


Figure 2.10: A Bézier curve and its control polygon. Image courtesy of Rogers (2000)

where t is the curve parameter and B_i are the control points. In figure 2.10, a Bézier curve and its control polygon is shown.

Bézier polynomials are somewhat constrained in the number of points that they may approximate without the degree of the curve becoming inconveniently high. A generalization of the Bézier approach known as B-splines overcome this problem. B-splines uses blending functions to combine the influence of a series of control or track points in an approximate curve.

Letting $P(t)$ be the position vector along the curve as a function of the parameter t, a B-spline curve is given by:

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t) \quad t_{min} \leq t < t_{max}, \quad 2 \leq k \leq n+1 \tag{2.3}$$

where the B_i are the position vectors of the $n + 1$ control polygon vertices, and the $N_{i,k}$ are the normalized B-spline basis functions. For the ith normalized B-spline basis function of order k (degree $k - 1$), the basis functions $N_{i,k}(t)$ are defined by the Cox-de

Boor recursion formulas. Specifically

$$N_{i,1}(t) = \begin{cases} 1, & \text{if } x_i \leq t < x_{i+1} \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

and

$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{2.5}$$

The values of x_i are elements of a knot vector.

A generalization of both Bézier curves and the B-spline curves are NURBS curves. A NURBS curve is defined by its order, a set of weighted control points and a knot vector. The primary difference between NURBS, Bézier and B-splines is the weighting of the control points which makes NURBS curves rational. B-splines are a special case of of rational NURBS where each control point has a weight of 1, that is regular and non-homogeneous. Currently, NURBS curves and surfaces are the standard for curve and surface description in computer graphics.

NURBS curves are given as

$$P(t) = \sum_{i=1}^{n+1} B_i^h N_{i,k}(t) \tag{2.6}$$

where B_i^h are the control polygon vertices for the non rational four-dimensional B-spline curve. $N_{i,k}(t)$ is the non rational B-spline basis function as given in equation 2.5.

Surfaces work in the same way as curves, only in two directions. These are called "tensor product" surfaces. As a result, instead of one parameter, there are now two, usually called u and v. The general formula for Bézier surfaces are:

$$P(t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \frac{n!}{i!(n-i)!} \frac{m!}{j!(m-j)!} (1-u)^{n-i} u^i (1-v)^{m-j} v^j B_{ij} \tag{2.7}$$

B_{ij} are the vertices of a polygonal control net. By comparing with the equation of

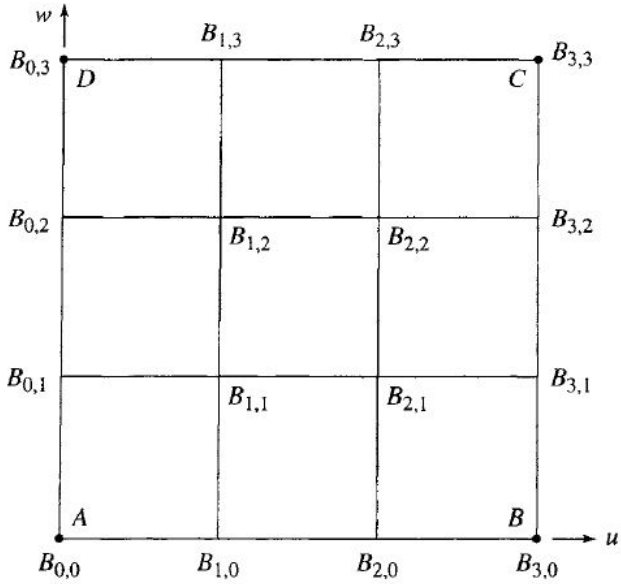


Figure 2.11: Schematic of the control net for a 4*x*4 Bézier surface. Image courtesy of Rogers (2000)

a Bézier curve given in equation 2.2, it can be seen that they are quite similar. The difference is that the equation of a Bézier surface contains two Bézier curve equations. The control net of a 4*x*4 Bézier surface is shown schematically in figure 2.11. Here, the v direction is given as w. Figure 2.12 shows several bi-cubic Bézier surfaces and their control nets. The figure shows how changes in the tangent vector affects the surface. B-spline surfaces and NURBS surfaces work in the same way as Bézier surfaces, but with different basis functions.

2.2.2 The STEP standard

In design and manufacturing, there are many different systems used to manage technical product data. Each system has its own data format to represent the same information. As a result, the same information has to be entered multiple times into multiple

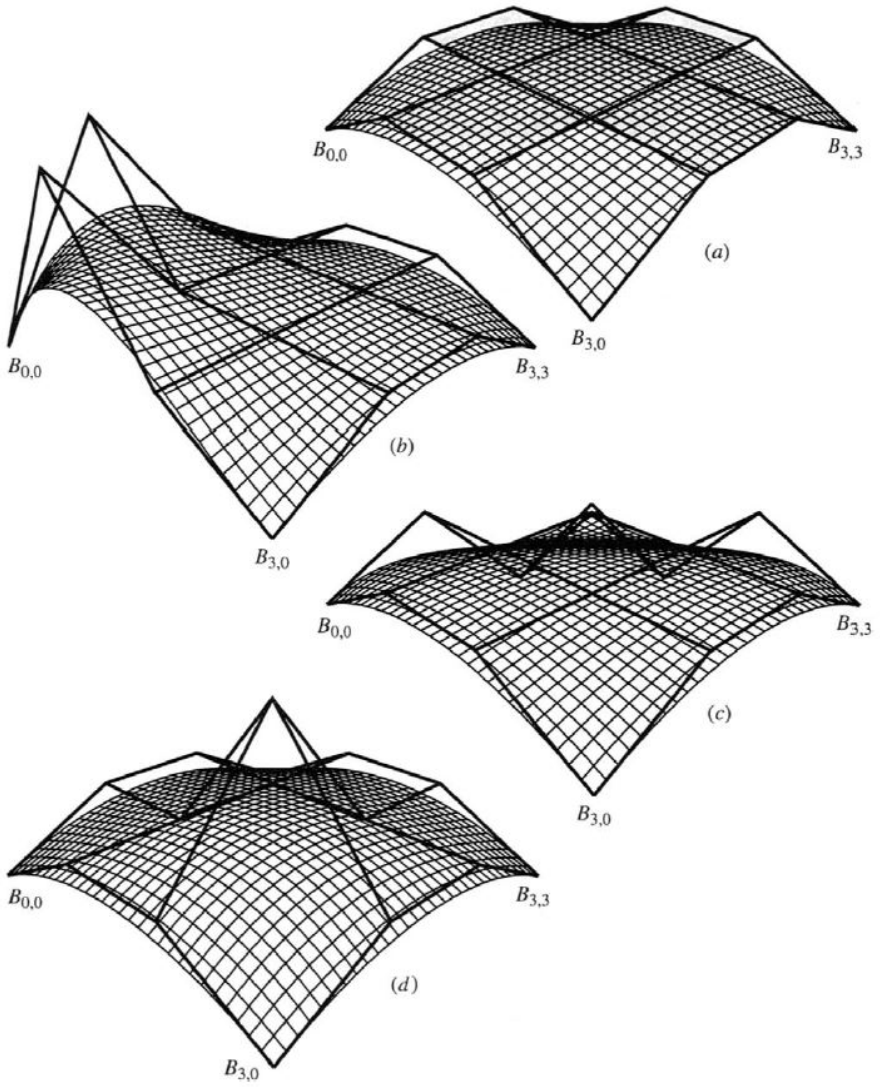
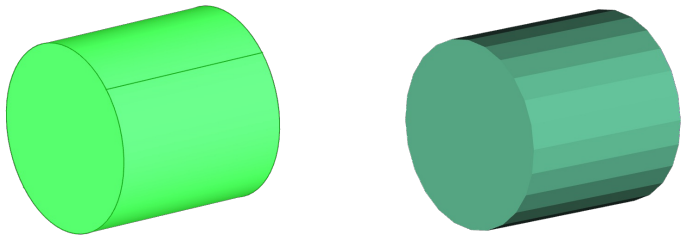


Figure 2.12: Bi-cubic Bézier surfaces. (a) Base surface; (b) effect of a change in both tangent vector magnitudes at $B_{0,0}$; (c) effect of a change in tangent vector direction at $B_{0,3}$; (d) effect of a change in twist vector magnitude at $B_{0,0}$. Image courtesy of Rogers (2000)

systems resulting in errors. For 3D design, this can become a large problem due to its complexity. To solve the problem, standards for data exchange was made. In the beginning, there were several different standards before an unifying effort was started by International Standards Organization (ISO) to create one international standard. The standard was named Standard for the Exchange of Product Model data (STEP), more formally known as ISO 10303, *The STEP Standard* (2018). STEP is represented using B-rep.

2.2.3 Stereolithography

STL was developed as a graphics exchange standard. Today, it is the most commonly used format for 3D-printing. It approximates the surfaces of the model using polygons, Chua et al. (2003). The number of polygons necessary to approximate a surface depends on the structure of the surface. Highly curved surfaces must employ many more polygons than a flat surface. The accuracy of the STL model is also determined by the size and number of the polygons. It is important to notice that the size of the file becomes larger as the number of polygons increase which creates a trade off between size, accuracy and print time. In figure 2.13, a 3D model of a cylinder is shown in both STEP and STL format. It can be seen that in the model represented using STEP, figure 2.13a, the surface of the cylinder is smooth, continuous and put together by different sub objects. In figure 2.13b, the cylinder is in STL format. It can be seen that the surface of the cylinder is approximated using polygons.



(a) Cylinder in STEP format (b) Cylinder in stl format

Figure 2.13: 3D model of cylinder

2.2.4 FreeCAD

CAD software enables engineers to design, inspect and manage projects with an integrated graphical user interface (GUI) on a computer. Some popular CAD software are Autodesk, Solidworks and FreeCAD. FreeCAD is a parametric 3D modeler made primarily to design real-life objects of any size, *FreeCAD* (2018). Parametric modeling allows you to easily modify your design by going back into your model history and changing its parameters. FreeCAD is open-source and highly customizable. It can be used interactively, or its functionality can be accessed and extended using the Python programming language.

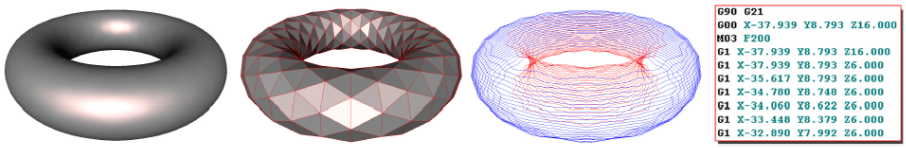


Figure 2.14: Typical process planning phases in AM: a design model (left) is tessellated to enter the Process Planning phase (centre-left). Such a tessellation is sliced (centre-right), and each slice is converted to a sequence of machine instructions (right). Image courtesy of Livesu et al. (2017).

2.3 AM processing pipeline

The first step of AM is to have a 3D model of what you want to print. Such a model is typically realized using CAD. When you have a 3D model, the next step is the process planning. Process planning is the sequence of operations required to move from design to realization. In AM, the phases of process planning is typically to tessellate the design model, to slice the tessellated model and to finally convert each slice into machine instructions, Livesu et al. (2017). A figure of the process going from a CAD model to machine instructions is shown figure 2.14.

2.3.1 Tessellation of model

The first step in process planning is to tessellate the 3D model. This is done by casting the CAD model into a corresponding Computer Aided Manufacturing (CAM) representation. A typical CAM file format is STL as described in section 2.2.3. STL is a triangular surface representation without representation of color, texture or other common CAD model attributes, Chua et al. (2003). The casting from CAD to CAM can be done within most software for 3D design such as Solidworks.

2.3.2 Slicing algorithm

After casting the design model into a CAM representation, the next step of the process planning is slicing. Slicing means dividing the geometry into a set of contours. Tradi-

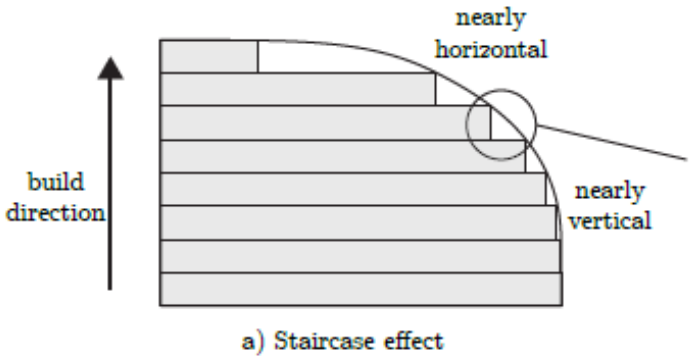


Figure 2.15: Staircase effect. Image courtesy of Livesu et al. (2017)

tionally this has been planar contours, but there has been some research on curved contours as well.

Planar contours

When slicing into planar contours it is commonly along the z axis aligned with the height of the model. Then each slice is a plane intersecting the shape at a given height. The simplest approach to divide the model into slices is to subdivide it uniformly, that is with a constant step in the z direction. Given a manufacturing layer thickness τ and the height of the model H, the model is divided into $N = \frac{H}{\tau}$ slices. Each slice i is then located at height $z_i = \frac{i+0.5}{N}$, Livesu et al. (2017).

One of the main drawbacks of planar contouring is the staircase effect, Livesu et al. (2017). This occurs as a result of the difference between the contours and the model. The staircase effect is shown in figure 2.15. As can be seen in the figure, the staircase effect is almost negligible when the geometry is close to vertical. However, when the geometry is near horizontal and highly curved, the staircase effect is large causing a large error between geometry and print. One solution is to reduce the step length, but this increases both the build time and cost. An alternative solution to

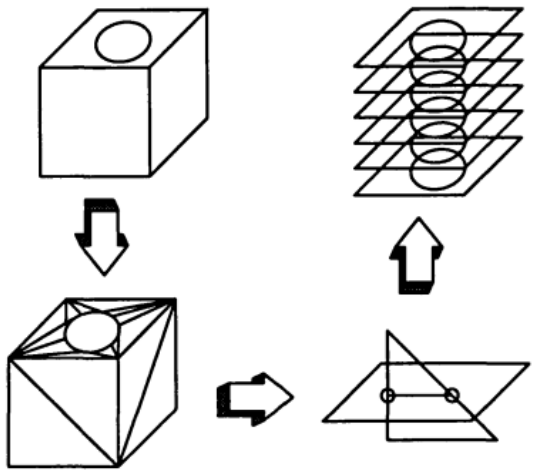


Figure 2.16: Slicing of triangle mesh process. Image courtesy of Kirschman and Jara-Almonte (1992)

reduce the staircase effect is to use adaptive step length in the z direction. Adaptive slicing approaches exploit this by adapting the thickness of each slice to the shape geometry, Sabourin et al. (1996). Thinner slices can then be used for curved geometries and thicker slices for close to vertical geometry.

After determining the set of slices, each slice plane has to be intersected with the input geometry. One way to do this is slicing of triangle meshes. An algorithm of this is described by Kirschman and Jara-Almonte (1992). The algorithm works by extracting, for each z layer, all intersections segments between the slice plane and the triangles of the STL file followed by forming loops. This process from design model to slices is shown in figure 2.16. There are many different implementations of this method and they mainly differ by how segments and loops are formed.

Direct Slicing

To avoid triangle mesh, several techniques has been developed to extract contours directly from the initial model. These methods are called direct slicing. In Jamieson

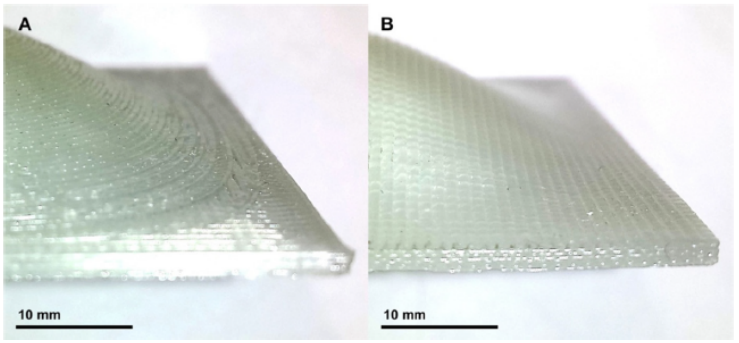


Figure 2.17: Images of sloped regions of simple example manufactured: (A) component manufactured using conventional layered contouring; (B) component with skin layer manufactured using curved contouring. Image courtesy of Allen and Trask (2015)

and Hacker (1995), the arguments for and against direct slicing is discussed. One argument for direct slicing is that for models with cylindrical shapes, a tessellated file will make the print inaccurate and most likely require hand finishing to obtain a smooth surface. Another advantage is that the slices are fed directly to the printer reducing pre-processing time and file size. One file format for direct slicing is Common Layer Interface (CLI). The drawbacks of direct slicing is that supports cannot be easily added to the nested sections and the ability to re orientate the model is lost. It is argued that the disadvantages is in fact benefits since the designer should design the models for AM to gain the full potential of the technology.

Research has been done on applying direct slicing on point clouds which are often obtained by 3D scanners or vision algorithms. For point clouds, triangle contouring is not applicable since the connectivity and topology is unknown. Yang et al. (2011) present an approach on slicing point-set into planar contours for AM use.

Curved contours

As already mentioned, one of the biggest drawbacks of planar contours is the staircase effect. This problem occurs especially on curved surfaces. To minimize that effect and also improve the mechanical properties of the part, there has been a lot of development

and research on AM using curved contours by Huang and Singamneni (2012), Allen and Trask (2015), Lim et al. (2016) and as mentioned in section 2.1.2 by Zhang et al. (2015), Zhang et al. (2016) and Alsharhan et al. (2017). Huang and Singamneni (2012) and Allen and Trask (2015) presents curved contouring and methods for developing a tool path for specific models. Lim et al. (2016) discuss the advantages and problems of curved contouring in addition to developing a novel method for generating a tool path for a broad specter of models. The main disadvantages of planar contours is the main advantages for curved contours. Firstly, the staircase effect is reduced. Secondly, the top surface is covered by one layer instead of multiple layer edges. Thirdly, the mechanical and aesthetic properties of extrusion based AM are improved. This is tested and proved by Alsharhan et al. (2017). The main problems with curved contouring is that it is necessary with more DOF for the printer, making the path generation more complicated. Also, a need to secure collision avoidance with the print is necessary. In figure 2.17, the result from the experiment performed by Allen and Trask (2015) is shown. Here it can be seen that the structure of the surface is greatly improved by using curved contouring instead of layered contouring for curved geometries.

2.3.2.1 Multi-directional slicing

Most 3D slicing methods within WAAM have focused on minimizing the need for support structures. One reason for this is that for metal components, the supports are normally deposited using the same material. This results in wastage of material, and the removal of such supports requires costly post-processing, Ding et al. (2016). A method to reduce the need for support structures is multi-direction slicing as presented in Ding et al. (2016). The method works by first decomposing the CAD model into sub-volumes before the sub-volumes are sorted using a depth-tree structure based on topology information for slicing. In figure 2.18, a model of the process is shown.

2.3.3 Support structures

After the slices and contours are ready, support structures has to be added to the part. Support structures are essential in AM, especially for extrusion based methods.

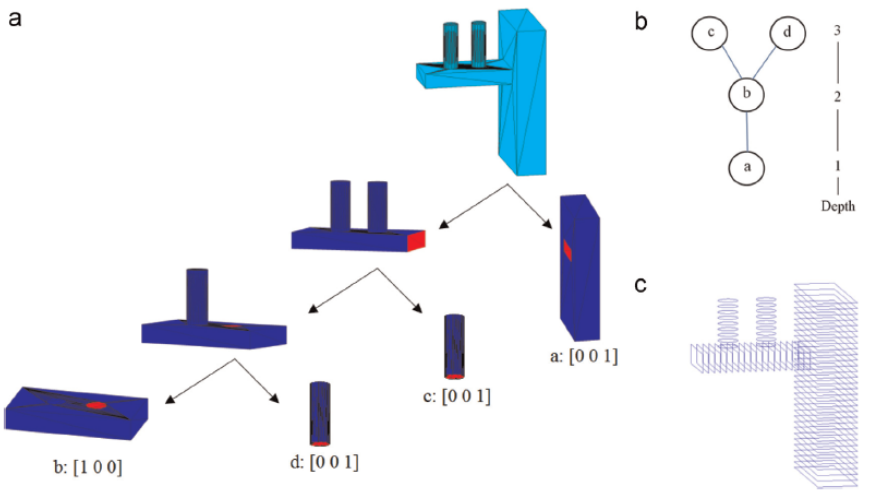


Figure 2.18: (a) a progressive decomposition of an example into sub-volumes with their own build directions; (b) sub-volume grouping with depth-tree structure; (c) model slicing along multiple build directions. Image courtesy of Ding et al. (2016)

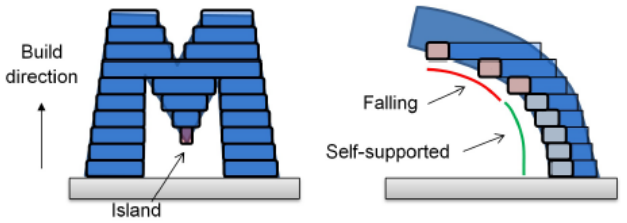


Figure 2.19: The letter M decomposed into layers. The figure to the left is showing an island and the figure to the right is showing collapsing print due to overhang. Image courtesy of Livesu et al. (2017)

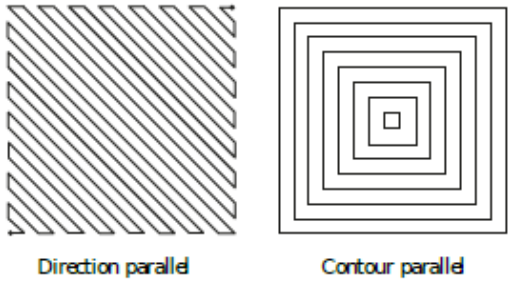


Figure 2.20: Direction parallel and contour parallel infill patterns for AM. Image courtesy of Livesu et al. (2017)

They are used to compensate for limitations of the manufacturing processes. Some limitations are certain geometrical structures such as overhang angles and islands. This is represented in figure 2.19. Another reason for support structures are to avoid printing the inner volume of the part to safe both time and cost. Support structures are split into two categories, internal and external support. The external support structures are typically printed in a different material than the rest of the print to facilitate part cleanup and removal, Gibson et al. (2010). This does however often require human intervention and is therefore a time-consuming and expensive step.

Internal support modify the inside of the object to achieve a trade-off between material cost, print time and physical properties, Livesu et al. (2017). They can be split into two main categories: techniques that create large empty pockets within object and techniques that create dense infill. The techniques that create large empty pockets within the object focus on saving both material and time. Within the area of dense infill there are different filling patters including direction parallel (or zig-zag) and contour parallel. The two dense infill methods are shown in figure 2.20

2.3.4 Machine instructions

There are several factors to consider when generating the tool path. In FDM it is very difficult to control the material deposition when a path begins and stops, making it

important to reduce the number of disconnected paths. Furthermore, long curvature paths are preferred over sharp turns, allowing higher nozzle speed. One final factor to consider is to reduce the airtime, the time necessary to move the nozzle from the end of a curve to the beginning of the subsequent one. This path planning problem has been shown to be related to the TSP, Wah et al. (2002). TSP is described in section 2.5.3.

2.3.5 AM software

There is a lot of AM software on the market for process planning. They generally work by uploading the CAM model you want to print and then return the machine instructions. Some of them are described, compared and rated on *17 best 3D slicer software tools for 3D printers* (2017). The different software can vary on price, who it is intended for and on operating system. Two free solutions are Ultimaker Cura and CraftWare. Ultimaker Cura is developed by the 3D printer company Ultimaker, but has its roots in open source. CraftWare is developed by CraftUnique to support their 3D printer CraftBot, but can be used for other printers as well. Both Cura and CraftWare has the option of switching between "easy" and "expert" mode depending on the experience and need of the user. The output from both programs is G-code in the file format .gcode.

G-code

G-code is the common name for the most widely used numerical control programming language. Smid (2003) defines numerical control as an operation of machine tools by the means of specifically coded instructions to the machine control system. Numerical control is is used in Computer Numerical Control (CNC). CNC refers to a computer "controller" that reads G-code instructions and drives the machine tool, *G Code* (2017). G-code instructions are preparatory commands. They preset and prepare the control system to a certain desired condition, mode or state of operation. Examples of G-code commands used in AM is shown in the table 2.1.

Command	Meaning
G0	Rapid positioning
G1	Linear interpolation
G28	Return to home
G90	Absolute distance mode
G92	Offset coordinate system and set parameters

Table 2.1: Commonly used G-code commands

2.4 Robot control

The development of industrial robots is characterized by the fusion of a large spectrum of multidisciplinary technologies. Automotive industries and their supply chains are the dominating customers for industrial robots today. This means that their needs and requirements has been driving the robot development. In order to find other industries that might drive future robot development, low cost safe robot systems that are easy to install, configure, calibrate, program and maintain are necessary, Brogårdh (2007). The complexity of programming remains one of the major hurdles preventing other industries from using industrial robots.

There are two main methods for robotic programming; online programming including lead-through and walk-through and Offline Programming (OLP). For online programming, a teach pendant is used to manually move the end effector to the desired position and orientation at each stage of the robot task. The relevant robot configurations are recorded by the robot controller and the robot program is generated. The advantages of this method is that it is intuitive, requires low programming skills and has low initial cost. The drawbacks however, is that it is limited by the skills of the operator and once the program is generated, it it very difficult to make changes, Pan et al. (2012). OLP is based on the 3D model of the robot system. It is more reliable and flexible compared to online programming. The drawback is that it is more complex, requiring higher programming skills.

Due to the complexity of OLP, numerous graphical software environments are created by robot vendors as well as non robot vendors. Almost every robot vendor

has their own platform and each OLP is made compatible with its corresponding robot hardware. ABB has RobotStudio, which is the most popular one, and KUKA has KUKA-sim and KUKA CAM-Rob.

Three environments developed by non robot vendors are Grasshopper 3D, Robotmaster CAD/CAM and Robot Operating System (ROS). Grasshopper 3D is an environment than runs within the Rhinoceros 3D CAD software, *Grasshopper* (2017). Grasshopper uses a visual scripting tool, allowing the user to create parametric structures without knowing how to write scripts. HAL is a robot programming add-on for Grasshopper that supports both ABB, KUKA and Universal Robots, *HAL Robotics* (2017). Robotmaster CAD/CAM integrates OLP, simulation and code generation. It is designed for users without deep knowledge in the field by using visualization of issues and opportunities to obtain the optimal robot program, *Robotmaster CAD/CAM* (2017). ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust behaviour across a wide variety of robotic platforms, *ROS* (2018).

There are no standardized robot control system either. KUKA's control system is based on x-y-z coordinates for translation and Roll, Pitch, Yaw to define the rotational angles, *KUKA* (2017). The programming language is called KRL. The control system of ABB, on the other hand, is also based on x-y-z coordinates for translation, but uses quaternions for rotational angles. ABB's programming language is called RAPID, *ABB Robotics* (2017). Each program or code is made specifically for one robot with specific configurations. As a result, it is not straight forward to run the same code on different robots with different configurations.

2.4.1 RobotStudio

RobotStudio is the OLP environment developed by ABB. It has tools for training, programming and optimization of robot systems. It has a virtual controller with the exact copy of the real software that runs on ABB robots, *RobotStudio®* (2017). This means that a system or code that runs successfully on the simulator in RobotStudio will also run successfully on the physical robot.

RAPID

RAPID is the programming language used to program ABB's industrial robots. The robot rely on three different types of motion, MoveL, MoveJ and MoveC, ABB (2007). MoveL is linear motion that forces the robot to move on a straight line between two points. This is the command that is used the most. MoveJ is a point to point motion that lets the robot move to the second point in the easiest way. This is typically used to move to the start point or the other side of the workspace. MoveC is arc motion defined by two points. The robot will then start from the point where the last command ended and create an arc through the first until the second point.

2.4.2 Robot Operating System

As mentioned, ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust behaviour across a wide variety of robotic platforms, *ROS* (2018). ROS is open source and written in C++ and Python. It can run on two operating systems, Linux and MacOS. One program in ROS is ROS-Industrial. This contains libraries, tools and drivers for industrial hardware. One feature of ROS-Industrial is a package that provide communication with ABB industrial robot controllers. The models for supported manipulators are associated with another feature in ROS-industrial, namely MoveIt!.

MoveIt!

MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation. It provides an easy-to-use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, R&D and other domains, *MoveIt!* (2018). MoveIt! is interfaced on over 65 robots by the community. A list of the robots can be found on their web page. If desired, it is possible to implement MoveIt on any robot. At the Norwegian

University of Science and Technology (NTNU), MoveIt is implemented on the KUKA robotics laboratory at the Department of Cybernetics.

2.5 Algorithms

This section is on based chapter 15, 16 and 32 in Cormen (2009).

2.5.1 Dynamic programming

Dynamic programming solves problems by combining the solutions to subproblems. It applies when the subproblems overlap - that is, when the subproblems share subsubproblems. In this context, a divide-and-conquer algorithm does more work than necessary, repeatedly solving the common subsubproblems. A dynamic-programming algorithm solves each subsubproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time it solves each subsubproblem. Dynamic programming is typically applied to optimization problems. Such problems can have many possible solutions and the goal is to find a solution with the optimal (maximum or minimum) value.

2.5.2 Greedy choice algorithms

For many optimization problems, using dynamic programming to determine the best choice is overkill; simpler, more efficient algorithms will do. A greedy algorithm always makes the choice that looks best at the moment. That is, it makes the locally optimal choice in the hope that this choice will lead to a globally optimal solution. It is important to notice that greedy algorithms do not always yield optimal solutions.

In order for a greedy algorithm to solve a particular optimization problem, the greedy-choice property has to be satisfied and the problem must have optimal substructure. The greedy-choice property is satisfied if we can assemble a globally optimal solution by making locally optimal (greedy) choices. In other words, we make the choice that looks best in the current problem, without considering results from sub problems. This differ from dynamic programming where we make a choice at each step, but the choice usually depends on the solutions to sub problems. The choice made by a greedy algorithm may depend on choices so far, but it cannot depend on any future choices or on the solutions to sub problems.

A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions to sub problems. This property is a key ingredient of assessing the applicability of both dynamic programming as well a greedy algorithms. For greedy algorithms, this can be applied with a rather direct approach. All we need to do is argue that an optimal solution to the sub problem, combined with the greedy choice already made, yields an optimal solution to the original problem.

Weighted greedy choice algorithm

A weighted greedy algorithm is the exact same algorithm as the greedy choice algorithm with the exception of the weighting. The greedy algorithm makes the locally optimal choice in the hope that this choice will lead to a globally optimal solution. The weighted greedy algorithm does the exact same thing, only that some weight might effect what the locally optimal choice is. Weighting works by altering the calculations that is the basis for the greedy choice based on some weighting criteria. This weighting criteria will vary depending on the application.

2.5.3 Travelling Salesman Problem

The following paragraph is taken from Lawler et al. (1985) and not Cormen (2009) as the rest of this section. In combinatorial mathematics, a graph is a finite set of vertices, some pairs of which are joined by edges. A cycle in a graph is a set of vertices of the graph which is such that it is possible to move from vertex to vertex, along edges of the graph, so that all vertices are encountered exactly once, and that we finish where we started. If a cycle contains all the vertices of the graph, it is called Hamiltonian. The TSP for a graph with specified edge lengths is the problem of finding a Hamiltonian cycle of shortest length.

The TSP is described as a salesman must visit n cities. The salesman wishes to make a tour visiting each city exactly once and finishing at the city he starts from. The salesman incurs non negative integer cost $c(i, j)$ to travel from city i to city j, and the salesman wishes to make the tour whose total cost is minimum, where the cost is the sum of the individual costs along the edges of the tour. The TSP is NP-complete. A

NP-problem is a problem where no polynomial-time algorithm has yet been discovered, nor has anyone yet been able to prove that no polynomial time algorithm can exist for any one of them.

Chapter 3

Preliminary project

In the autumn 2017, a method for enabling AM using robot manipulators based on existing solutions was investigated in the authors specialization project with the title: "Additive manufacturing by robot manipulator: A method for generating robot trajectory based on CAD model using existing solutions". The method consisted of five main steps:

1. Cast the CAD model into a CAM model in STL format.
2. Generate 3-axis machine instructions based on the CAM model using AM software. In the project, CraftWare, *CraftWare* (2017), was used. The machine instructions were given in g-code. A picture of the result in CraftWare including the path and the machine instructions is shown in figure 3.1.
3. Convert the machine instructions into robot path using Machining PowerPac, *RobotStudio Machining PowerPac* (2017), in RobotStudio, *RobotStudio®* (2017).
4. Simulate the generated robot path using the virtual controller in RobotStuio.
5. Run the simulated path on the physical robot.

With some smaller modifications of the generated code, the robot managed to follow the generated path. It was, however, argued that the method has drawbacks

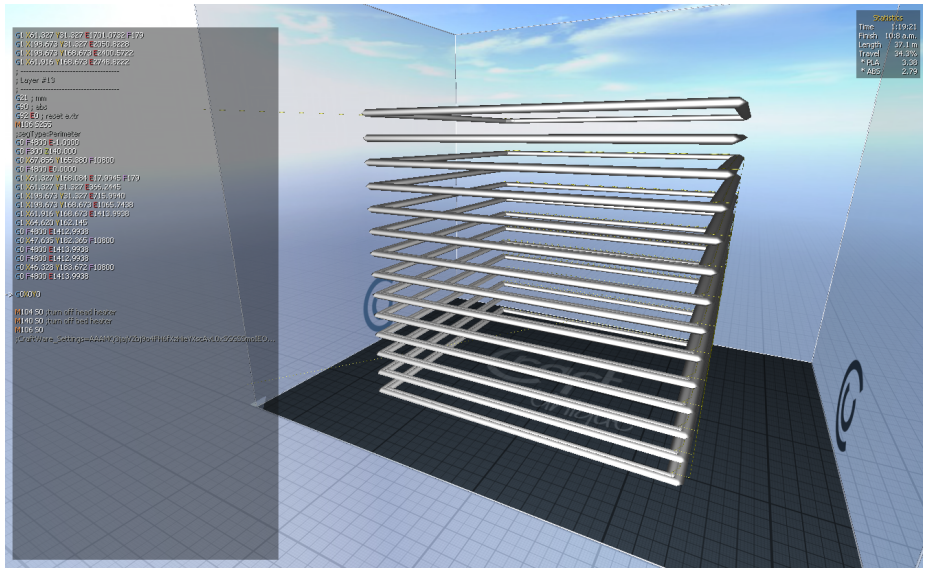


Figure 3.1: Result from the slicing process using CraftWare

and limitations that makes the method unfit for large-scale AM using a 6 DOF robot. One drawback was that the slicer tool did not generate an optimal path. The most significant drawback was, however, that the machine instructions is generated for a 3 DOF 3D printer meaning that the robot cannot take advantage of its 6 DOF for large-scale AM using this method. Some lost advantages is to use other approaches than bottom-up AM to reduce the need of support structures and to use non-planar contours to improve the aesthetic and material properties.

3.1 Further work

In the specialization project, one problem was that the used slicing tool, Craftware, had limited options in the advanced settings. As a result, it was impossible to generate slices without an inner wall. To further investigate the method for automatic generation of robot trajectories based on CAD models developed in the specialization project, it was decided to look for other slicing tools that had the desired possibilities. After testing various options, the choice fell on SelfCAD. SelfCAD is an online browser-based CAD/CAM platform which allows the user to model, sculpt, slice and print online, *SelfCAD* (2018).

In SelfCAD, it is possible to set a Custom FDM printer as your choice. This printer can be modified in whatever way you like. For this test, layer height was set to 2.0 mm, wall thickness to 4.0 mm, wall line count to 1 and top/bottom thickness to 0 mm. All support structures such as infill and brim was disabled. As in the project, the generated g-code was imported, converted and simulated using RobotStudio and its corresponding Machining PowerPac. The complete simulation of the print can be seen in figure 3.2 and 3.3. As can be seen in the close up picture of the simulation in figure 3.3, the path is without an inner wall as planned.

The result of this improved method is a fully functional robot trajectory that is generated based on a CAM model. It is simulated and verified using the powerful virtual controller in RobotStudio. This trajectory could have been tested on a physical robot with AM tools as it is.

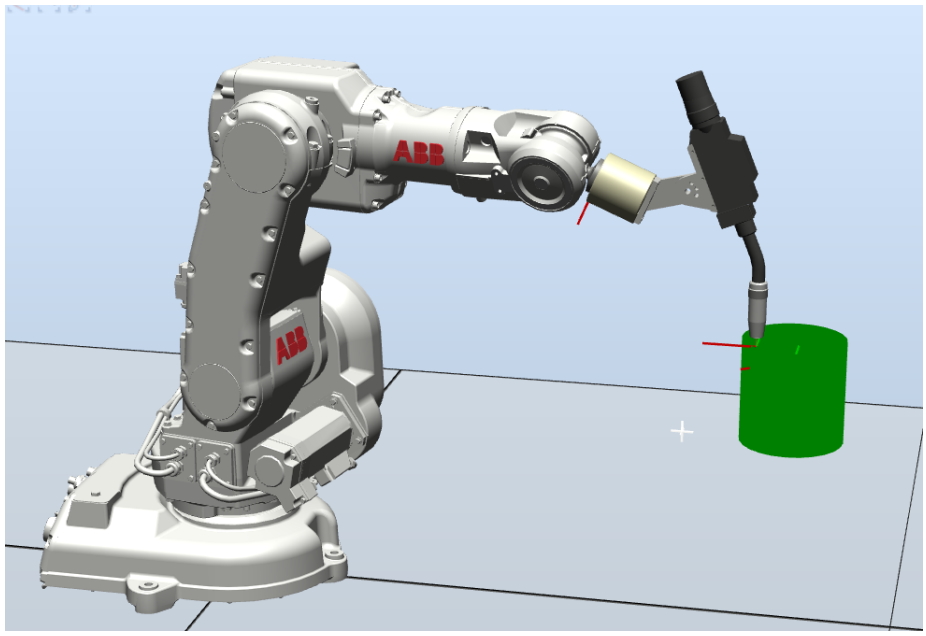


Figure 3.2: Result of simulation of printing a cylinder in RobotStudio

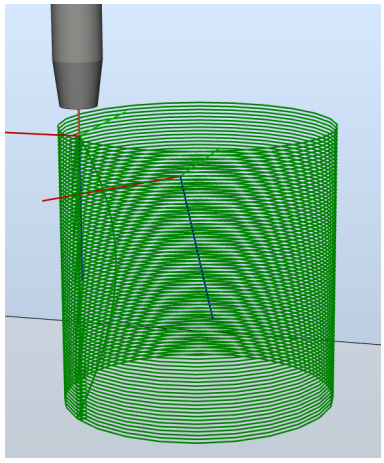


Figure 3.3: Close up result of simulation of printing a cylinder in RobotStudio

3.2 Future work

Even though a fully functional robot trajectory was generated as described in the previous paragraph, the method still has limitations. The most important limitation being that the method is based on solutions made for 3 DOF 3D printers, while the goal is to take advantage of the 6 DOF of a robot manipulator. Another limitation is that both the slicing software and Machining PowerPac converter are considered to be a processes that are difficult to control, meaning that the the user cannot control what happens besides the settings available in the user interface. One last limitation is that g-code for AM uses only straight line segments. This means that curved surfaces will be approximated by many small straight line segments. Based on the mentioned limitations, it was decided to find another approach to automatically generate paths for AM using robot manipulators. This will be discussed in the following chapters.

Chapter 4

AM Path generation

In chapter 3, a method for generating robot trajectories based on CAD models using existing solutions was presented briefly and discussed. Based on the result it was concluded that it was necessary with another method for automatic generation of paths.

4.1 Choice of solution

Below is a list of possible improvements for the new method:

1. Take advantage of the DOF
   (a) Curved contouring
   (b) Reduce the need for support structures
   (c) Multi-directional slicing
2. Avoid g-code
3. Improve control over the process
4. Use a CAD format instead of CAM

The first and most important improvement to consider is to take advantage of the DOF. This could be done in several ways including curved contouring or multi-directional slicing as described in section 2.3.2 or it could be to create a path that allows printing overhang without the need for support structures. The next two mentioned improvements is to avoid g-code and to improve control over the process. Both is described in chapter 3. If g-code is avoided, there is no longer necessary to convert the machine instructions into robot manipulator code. This also means that RobotStudio Machining PowerPac is no longer needed. This makes it easier to improve the control over the process which includes avoid using software where it is difficult to control the process. In the preliminary project, this was a large problem with both the AM software and RobotStudio Machining PowerPac. One more advantage of avoiding g-code and thus finding another way of representing the path is that there is more freedom in the choice of robot control environment. The final improvement was to use a CAD format instead of a CAM format such as STL which is as approximation of the CAD model. It was a goal to use a CAD format that was platform independent. CAD was presented in section 2.2 and here under STEP as a possible format. STEP is defined using boundary representation and is platform independent.

In section 2.3.2 and more specifically Alsharhan et al. (2017), a method using curved contouring is presented. One input to the method is the part geometry and orientation represented by the surface equation of the desired part. With surface equation as input, there is one main drawback of the method: the surface must be described using a surface equation. This means that more complex shapes that is represented using numerical geometry cannot be printed. Within surface finishing as presented in section 2.1.3, however, paths that follows curved contours is generated for complex shapes as well. The method described in Lin et al. (2015) generate a set of control points based on the CAD model and then generate a path based on these control points. In this particular paper, the path is generated using TSP to improve more established methods such as zigzag and SFC.

Based on the findings in the literature study as described above, it was decided to make an AM path generation system. As opposed to the specialization project, only the path planning problem will be investigated. The robot control problem including

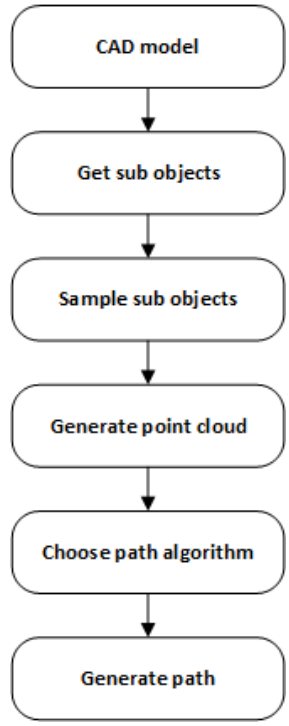
tool-path planning and collision avoidance is outside the scope of this thesis.

The path planning system will be inspired by the system used for surface finishing. This allows curved contouring, possibly printing in overhang, improving the control over the process and it allows the use of a CAD format instead of CAM. Since this is only a path planning system, using or avoiding g-code will not be a issue for this thesis.

4.2 AM Path generation system

The system is implemented using Python and ran as a Macro in FreeCAD. This allows free interaction between the CAD model and the script. The main idea behind the solution is to generate a point cloud of samples from the CAD model and then run a path generation algorithm on the point cloud. A flowchart of the system can be seen in figure 4.1. The first step is to have a CAD model in STEP format open in FreeCAD. Next, the desired sub object(s) are selected. From here on out in this thesis, a sub object is considered to be a face as described in section 2.2.1. By running the script on the selected sub object(s), the system will sample the sub object(s) separately before combining them into a point cloud. Before generating a path, an algorithm has to be chosen. The choices include TSP, greedy choice and weighted greedy choice. Finally, a path is generated based on the point cloud. The generated path and samples can be displayed in FreeCAD together with the CAD model to see the result. The system is implemented using four classes: SubObject, Sample, Path and PointCloud. An UML diagram of the system can be seen in figure 4.2.

Figure 4.1: Flow chart of the automatic path generation process.



4.3 Sampling

To be able to generate a path, it was necessary to sample the surface of the object. This was done by sampling each geometric surface separately before gathering all the samples in a joined point cloud.

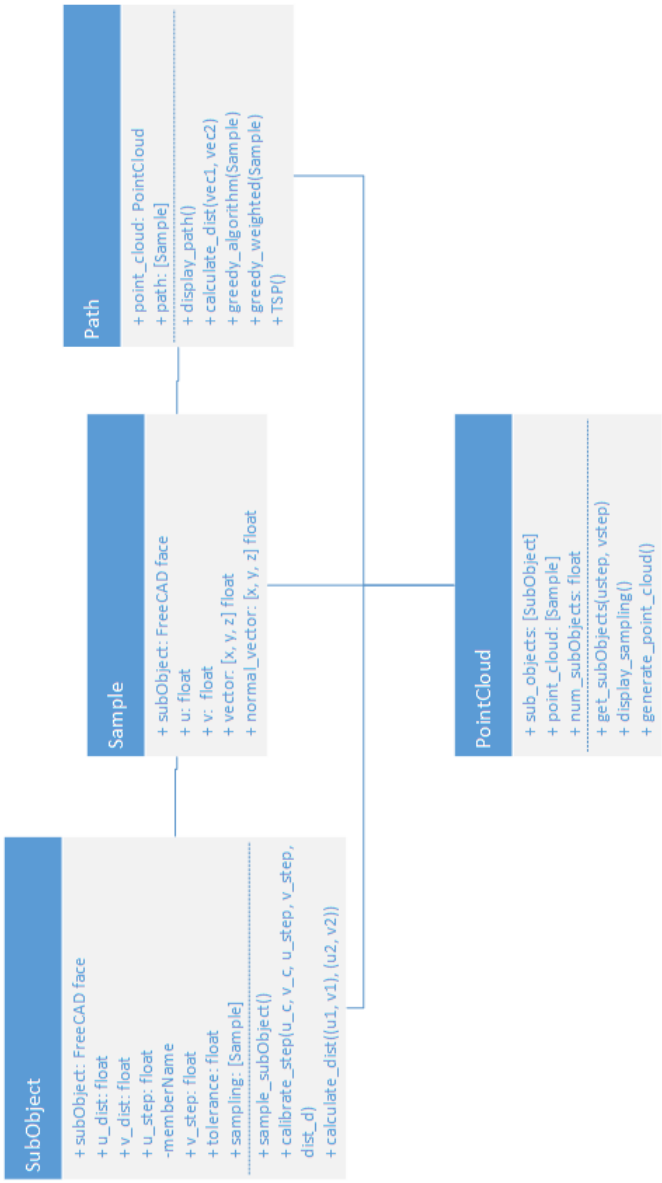


Figure 4.2: UML diagram of the automatic path generation system

4.3.1 Sample object

Each sample is stored as a Sample object. This object contains reference to the FreeCAD sub object the sample belongs to, the u and v parameter coordinates, the Cartesian coordinates and the normal vector as can be seen in the Sample class in figure 4.2. The normal vector is not relevant for this system, but for future work on tool-path planning it will be.

4.3.2 Sampling of sub object

A sub object, or face, is defined by two different coordinate systems in FreeCAD: Cartesian coordinates and parameter coordinates. All Cartesian coordinates for the entire CAD model are defined relative to the same reference point, and is given in the form (x, y, z). Parameter coordinates, however, are defined relative to a reference point that is unique for every sub object. It is given in the form (u, v).

The parameter coordinates (u and v) of a sub object is used for sampling by iterating over the face with a given step length. For each new iteration, it is checked if the sample is on the face by using the function `isInside` in FreeCAD. All valid samples are stored in a list of samples before the list is added to the point cloud. The sampling of each sub object is implemented in the SubObject class that can be seen in the UML diagram in figure 4.2.

When sampling a sub object, the user has to decide the desired distance (in millimetres) between each sample in both the u and v direction. A step length in both u and v direction is calculated based on the desired distances using calibration. This is described further in the following section.

4.3.3 Calibration of step

Since u and v are parameter coordinates, it was necessary to calibrate the step length in both the u and v direction to get the desired distance between each sample. For each iteration, the distance, d_k, between two points in the three-dimensional Euclidean

space using the current step is calculated using the formula

$$d_k = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{4.1}$$

where (x, y, z) is the Cartesian coordinate. $p_1 = (x_1, y_1, z_1)$ is found through the parameter coordinate (u_1, v_1) and $p_2 = (x_2, y_2, z_2)$ is found through the parameter coordinates $(u_1+\alpha_u, v_1+\alpha_v)$. The Cartesian coordinates is found through the parameter coordinates by using the function `valueAt` in FreeCAD. α_u and α_v is the step size in the u and v direction respectively. Since only one direction is calibrated at the time, either α_u or α_v will be zero during the calibration. The rest of the description is made independent of which direction that is calibrated.

The calibration used in this thesis is strongly related to line search which is a basic iterative approach to find a local minimum of an objective function. It is described in Nocedal and Wright (2006) accordingly:

> In the line search strategy, the algorithm chooses a search direction p_k and searches along this direction from the current iterate x_k for a new iterate with a lower function value. The distance to move along p_k can be found by approximately solving the following one-dimensional minimization problem to find a step length α:
>
> $$\min_{\alpha>0} f(x_k + \alpha p_k) \tag{4.2}$$

The implemented calibration works as follows. If the calculated distance is outside some area of tolerance, a,

$$a \leq d_k - d_d \leq -a \tag{4.3}$$

where a is given by

$$a = \frac{d_d}{t} \tag{4.4}$$

where t is the tolerance given by the user and d_d is the desired distance between the

two points. The step, α, will then be updated using the formula

$$\alpha_{k+1} = \alpha_k \frac{d_d}{d_k} \tag{4.5}$$

Pseudocode of the calibration function can be seen below.

```
def calibrate_step(points, step, desired_distance, tolerance)
  calibrated = False

  while not calibrated
    calculate distance between the points using current step

    if not calibrated and distance is outside area of tolerance
      update step
    else
      calibrated = True
  return step
```

Listing 4.1: Pseudocode for calibration of sampling.float

4.3.4 Generating point cloud

As mentioned, all the samples were gathered in a joined point cloud. The point cloud is a list of Sample objects which was described above. This point cloud can be displayed in FreeCAD with or without the corresponding CAD model. The point cloud is implemented through the PointCloud class which can be seen in the UML diagram in figure 4.2.

4.4 Path generation

After generating the point cloud, the next step of the process is to generate a path. Before the path can be generated, however, the goals and requirements of the path has to be established. Assumptions and/or simplifications of the problem also has to be taken into account.

4.4.1 Requirements of path

The requirements of the path can vary depending on the material. Different materials has variations in their ease of use and different physical properties. This means that one path can be feasible with one material and infeasible with another. Other requirements are due to practical reasons such as material extrusion.

In Ding et al. (2014), a list of requirements for a tool-path planning strategy for WAAM is given and described as follows:

1. Geometrical accuracy: as the resolution of arc welding is relatively low, the outlines of 2D geometries should be fabricated by contour patterns which could effectively improve the geometrical quality of the part.

2. Minimize the number of tool-path passes: the cumulative deviations introduced by the uneven weld bead geometry at the start and end portions of each welding pass will limit the maximum number of layers that can be added together before vertical build errors become problematic. Therefore, the number of welding passes should be minimized to reduce starting-stopping sequences within each layer. A continuous path is preferred here.

3. Minimize the number of tool-path elements: tool path elements are a series of line segments representing the travel path of the tool. In general, at the ends of tool-path elements, the wire feed rate should be adjusted to avoid a deposition error caused by a rapid change of tool-path travel direction. To improve surface accuracy, the number of tool-path elements should be minimized.

4. Simple algorithm with rapid implementation: the path planning algorithms should be simple and quick to implement to reduce the pre-processing computational time.

In the listed requirements for a path in WAAM, some are more relevant than others. One of the most relevant being that a continuous path is preferred over several smaller paths since it is difficult to control the extrusion of material during start and stop. Furthermore, it is preferred that the path is without self-intersections, again

since it is difficult to control material extrusion. Lastly, it is preferred that the path planning process is efficient in order to reduce the pre-processing computational time. As mentioned, the type of material affect the requirements. What to print also affect the path. Due to this, some assumptions had to be taken to confine the problem.

4.4.2 Assumptions

To continue making a path generation system, the problem had to be confined. This is done by making certain assumptions. The two assumptions are

1. Only simple surfaces
2. Fast curing material

Only simple surfaces means no solid models, only surfaces, and that the surfaces are relatively smooth. By assuming that a fast curing material is used, mid-air printing like achieved by Mataerial, presented in section 2.1.3, is possible. This enables printing both horizontally and upside down. In this way, overhang structures can be printed without the use of support structures.

4.4.3 Algorithms

Before choosing a path generation algorithm, it has to be established what properties the desired path has. Based on the requirements, certain conclusions can be drawn. It is desired that each sample is visited once and not multiple times. This means that the path is a-cyclic. Furthermore, since it is difficult to start and stop material extrusion, it is desired that the path is not self-intersecting. The distance between each sample can be thought of as weights. The distance, or weight, is however the same if you go from A to B as from B to A, making this an undirected problem with only positive weights. Based on these findings, it was decided to try out three different algorithms, including greedy choice, weighted greedy choice and Travelling Salesman Problem (TSP). All algorithms are implemented through the Path class that is shown in the UML diagram in figure 4.2.

Greedy choice algorithm

The greedy choice algorithm is described in section 2.5.2 as an efficient algorithm for optimization problems. It makes the locally optimal choice in the hope that this choice will lead to a globally optimal solution. In addition to being efficient, the greedy choice algorithm is also easy to implement. This is why this was the first algorithm to be implemented.

The pseudocode of the implementation is given in algorithm 1. It starts by choosing a starting sample. This is given by the user. Next, the distance to the other samples in the point cloud are calculated. Based on the distances, a greedy choice is made. Finally, the current sample is updated and added to the path. This loops until there is no unvisited samples left.

Algorithm 1 Greedy choice algorithm

```
1: Choose starting sample
2: while there is unvisited samples do
3:     Calculate distance to the other samples
4:     Make greedy choice
5:     Update current sample
6:     Add sample to path
7: end while
```

Weighted greedy algorithm

The greedy algorithm always chooses the sample that is the closest to the current sample. This might not always be the optimal choice. It could for example be beneficial if the path completed one level before moving on to the next. This can be controlled by adding weights in the calculations. In this way, the user can choose the starting point and printing direction that is most likely to give the best result.

The pseudocode of the implementation of the weighted greedy algorithm is given in algorithm 2. By comparing with algorithm 1, it can be see that they are very similar. The only difference between the two algorithms are the weight that is added on the distance if some weighting criteria is satisfied. Two different weighting systems was

implemented: u-weighting and y-weighting. u-weighting was implemented to affect the parameter coordinates u and v. A weight is added in the weighting algorithm if the u-value of the next sample is different from the current u-value. In this way, the path will complete one u-level before moving on to the next. v-weighting was not implemented in this thesis, but it would have worked in the exact same way. y-weighting was implemented to affect the Cartesian coordinates $x - y - z$. This works by adding a weight on the y-parameter of the distance calculations. Only y-weight was implemented in this thesis since it was the only one that was applicable as will be seen later. x and z could have been implemented in the exact same manner if found applicable.

Algorithm 2 Weighted greedy choice algorithm

```
1:  Choose starting sample
2:  while there is unvisited samples do
3:      Calculate distance to the other samples
4:      if weighting criteria is satisfied then
5:          Add weight on distance
6:      end if
7:      Make greedy choice
8:      Update current sample
9:      Add sample to path
10: end while
```

TSP

The TSP is described in section 2.5.3 as: the salesman wishes to make a tour visiting each city exactly once and finishing at the city he starts from, and he wishes to make the tour whose total cost is minimum.

Since TSP is a very complex algorithm, an already implemented solver was found at the web page that belongs to Joao Pedro Pedroso at the University of Porto, *TSP solver* (2018) [1]. Input to the solver is a series of 2D coordinates and the shortest path,

[1] This solver is a part of the programming code for the book "Metaheuristics: A programming guide", Kubo and Pedroso (2009)

Algorithm 3 TSP algorithm

1: Calculate distance matrix
2: Create greedy tour
3: Calculate length of tour
4: Local search starting from greedy tour

visiting all the coordinates, is returned. To verify that the solver worked, four simple test sequences was ran through the solver. The result of the four different sequences can be seen in figure 4.3. All paths are found using greedy choice on the initial tour and local search to improve the initial tour. Pseudocode of the solver is given in algorithm 3. Local search uses breadth first strategy until reaching local optimum. The blue dots are the points of the input sequence and the calculated shortest path is displayed in red. All axes are scaled to be equal in the figures. By reviewing the different paths, it can be seen that they are all indeed the shortest paths. It was therefore concluded that the solver worked satisfactory.

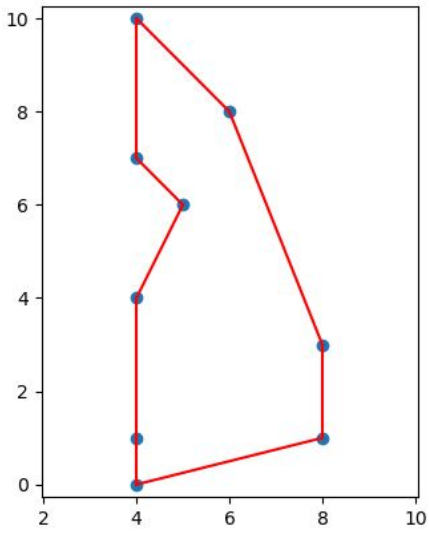
As mentioned, the input to the solver is a series of 2D coordinates. In the AM path generation problem, there points are, however, 3D coordinates. To make the solver applicable, some modifications was necessary. The solver makes all its calculations based on a distance matrix that is calculated in the beginning of the TSP algorithm. All distances in this matrix is calculated using the following formula

$$d_k = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.6}$$

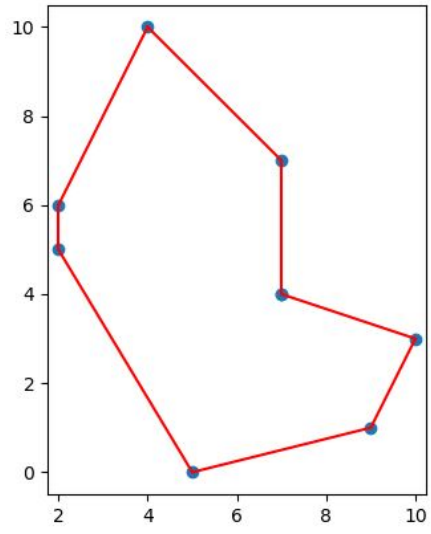
By changing this distance calculation from 2D to 3D, the solver was adapted to fit the AM path generation problem. The new formula was

$$d_k = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{4.7}$$

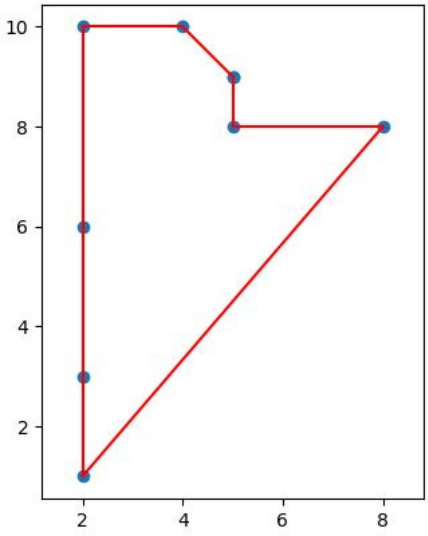
where (x, y, z) is the Cartesian coordinate.



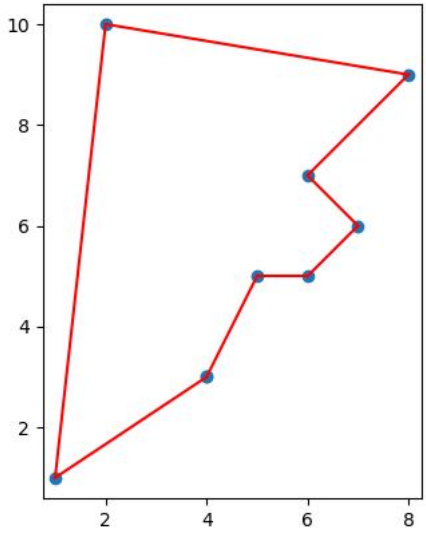
(a) TSP sequence 1



(b) TSP sequence 2



(c) TSP sequence 3



(d) TSP sequence 4

Figure 4.3: TSP solver tested on different input sequences

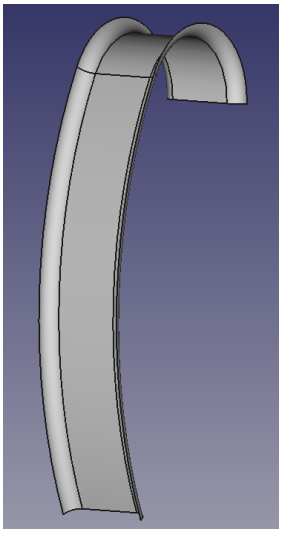
Chapter 5

Results

In chapter 4, the AM path generation system and implementation was presented. In this chapter, the results of testing the system is presented. First, the results of the sampling and point cloud generation is presented and next, the results of the path generation is presented including the results using the different algorithms.

For testing both sampling and path generation, two CAD models were used. The first model was of something resembling a shoehorn as can be seen in figure 5.1. This model was mainly chosen because of the overhang at the top of the model, but also because it is made up of several faces. The second model was of a propeller which can be seen in figure 5.2. Due to the complexity of propeller models, propellers today are mainly made using machining. It was therefore interesting to choose a propeller to work on for AM. The complex features include curving of the propeller blades and curved seams between the propeller blades and the center cylinder.

Figure 5.1: Shoehorn model used for testing the system



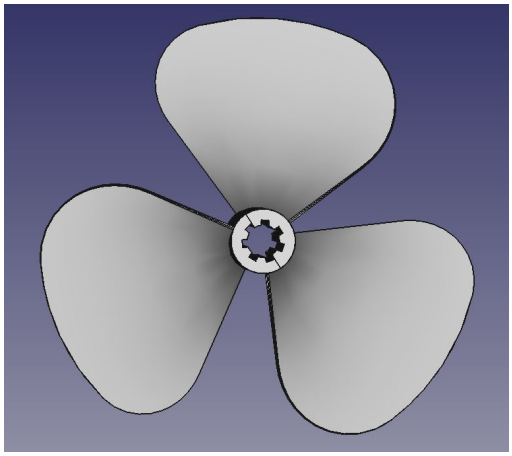


Figure 5.2: Propeller model used for testing the system

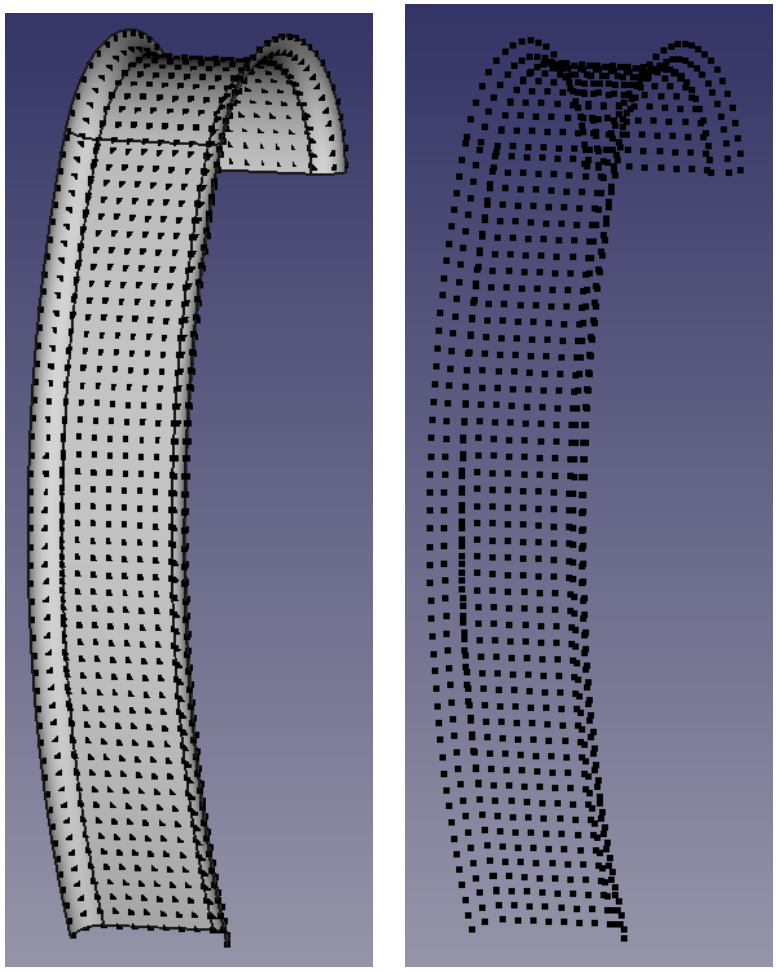
5.1 Sampling

In the AM path generation system, the first step of the process was to sample the surface of the selected sub objects of the CAD model. These samples was gathered in a point cloud. The process of sampling the sub objects and gathering the samples in a point cloud is described in section 4.3.

5.1.1 Shoehorn model

The first model that the sampling was tested on was the shoehorn model. The shoehorn model consists of six combined faces. For the first test, the step length was only calibrated once for each sub object resulting in non-adaptive sampling. In table 5.1 and figure 5.3, the result of the non-adaptive sampling is given. The sampling in the u and v direction was both set to 10.0 mm and this resulted in 961 samples. As can be seen in figure 5.3, the entire model was sampled with about the same distance between each sample. By studying the sampling closely, it can be seen that the sampling between different faces is not optimal.

In hope to fix the sampling error between faces, adaptive sampling was imple-



(a) Point cloud of non-adaptive sampling of shoehorn together with CAD model

(b) Point cloud of non-adaptive sampling of shoehorn model

Figure 5.3: Non-adaptive sampling of shoehorn model

Non-adaptive sampling of shoehorn model	
Distance between samples (u, v)	10 mm x 10 mm
Number of samples:	961

Table 5.1: Non-adaptive sampling of shoehorn model

Adaptive sampling of shoehorn	
Distance between samples (u, v)	10 mm x 10 mm
Number of samples:	960

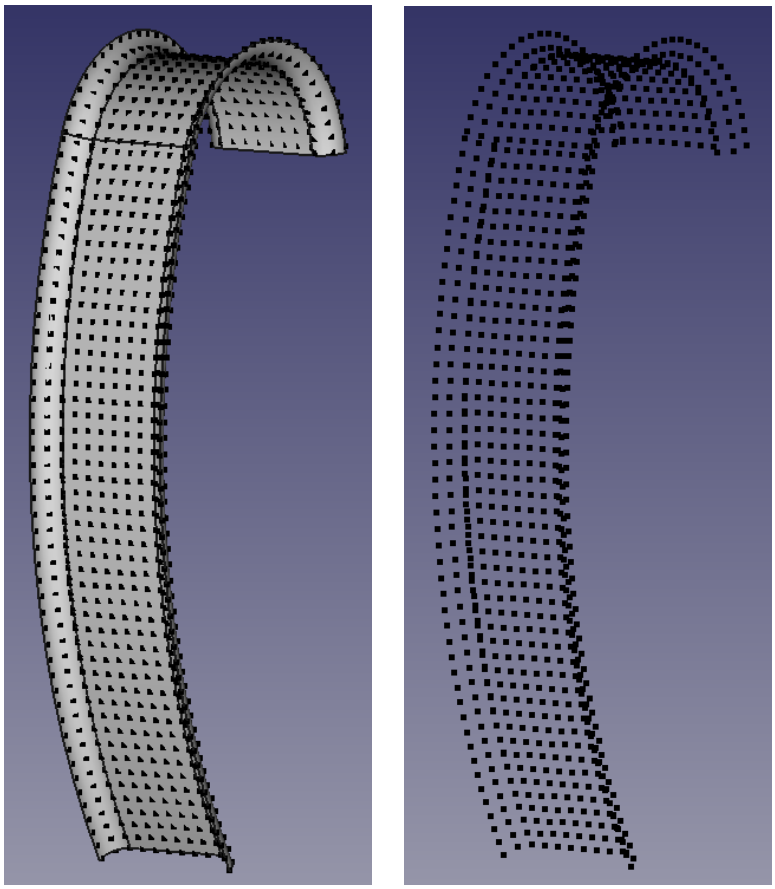
Table 5.2: Adaptive sampling of shoehorn model

mented. Adaptive sampling in this context means calibrating the step length for each iteration. The result of sampling the shoehorn model with the new calibration method is given in table 5.2 and figure 5.4. The same desired step length in the u and v direction was used and this resulted in 960 samples, one less that without adaptive sampling.

By comparing figure 5.3 and 5.4 it can be seen that they are more or less identical, both with the same sampling problem between faces. In figure 5.5, a close up picture of the sampling error is given. The close up photo shows that there is an error between the samples in the seam of the faces. This is almost negligible at the bottom of the figure, the start of the sampling, but grow larger further up along the model. After looking closer into the problem, it was found that the u and v direction is defined in different directions for the two largest faces in the middle and the thinner faces on the sides. This result in the sub objects being sampled in the same direction by considering u and v coordinates, but different directions in Cartesian coordinates.

5.1.2 Propeller blade model

The second model the sampling was tested on was the propeller model. More specifically it was tested on the propeller blade which is one the sub objects. Like with the shoehorn model, the sampling was tested with the same desired step length in both u and v direction and the calibration on the step length was only performed once at the



(a) Point cloud of adaptive sampling of shoehorn model with CAD model

(b) Point cloud of adaptive sampling of shoehorn model without CAD model

Figure 5.4: Adaptive sampling of shoehorn model

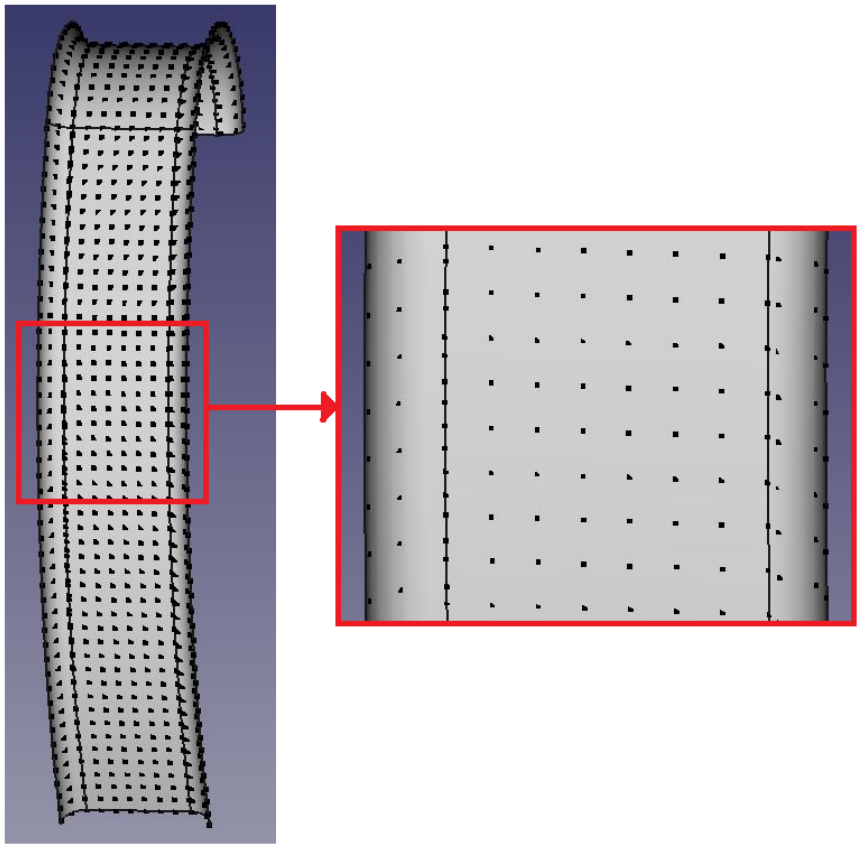
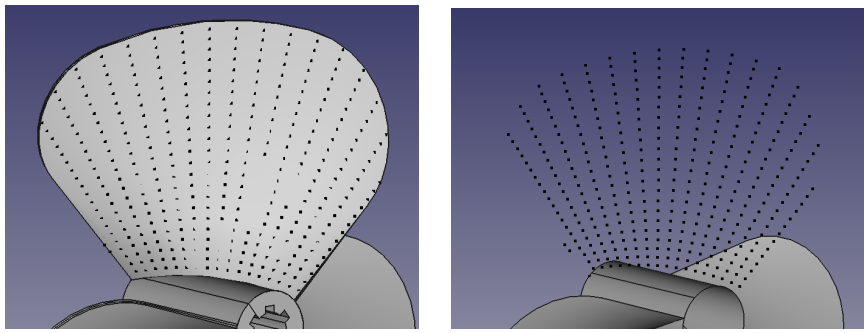


Figure 5.5: Sampling error displayed on sampling of shoehorn

Non-adaptive sampling of propeller blade	
Distance between samples (u, v)	10 mm x 10 mm
Number of samples:	372

Table 5.3: Non-adaptive sampling of propeller blade



(a) Point cloud of non-adaptive sampling of propeller blade together with CAD model

(b) Point cloud of non-adaptive sampling of propeller blade

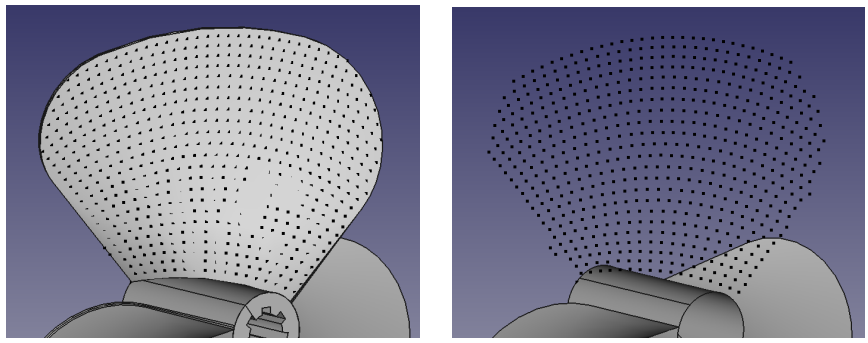
Figure 5.6: Non-adaptive sampling of propeller blade

beginning of the sampling. The result of the non-adaptive sampling can be seen in table 5.3 and figure 5.6. The distance between each sample was set to 10.0 mm x 10.0 mm and the number of samples was 372. As opposed to the result from the shoehorn sampling, the distance between each sample here is not the same for the entire face as supposed to. It can also be seen in the figures that the sampling follows the curvature of the seam between the propeller blade and the cylinder in the center.

In order to get the same distance between all samples on the propeller blade, adaptive sampling was tested. As for the shoehorn test, the same desired step length was used. The result from the adaptive test can be seen in table 5.4 and figure 5.7. With adaptive sampling, the number of samples was 609 which is a large increase from 372 with the non-adaptive sampling. By comparing the figure with non-adaptive sampling, figure 5.6, with the figures with adaptive sampling, figure 5.7, it can easily be seen that there are more samples with adaptive sampling and that the distance between each

Adaptive sampling of propeller blade	
Distance between samples (u, v)	10 mm x 10 mm
Number of samples:	609

Table 5.4: Adaptive sampling of propeller blade



(a) Point cloud of adaptive sampling of propeller blade with CAD model

(b) Non-adaptive sampling of propeller blade without CAD model

Figure 5.7: Point cloud of adaptive sampling of propeller blade

sample is constant.

5.2 Path generation

After the point cloud was generated, the next step of the AM path generation process was to generate the path. This was done using path generation algorithms. Three different algorithms were tested: greedy choice, weighted greedy choice and TSP. Based on the result from section 5.1, the path algorithms were only tested on the point clouds generated using adaptive calibration. To be able to compare the results, the desired distance between samples was set to 10.0 mm x 10.0 mm in u and v direction respectively. The starting point of all algorithms is set to the first point in the point cloud. The result from the three path algorithms is presented in the following sections.

In section 5.1.1, the result of sampling the shoehorn model was presented including the sampling error that occurs between faces. This sampling error is present when generating paths for the shoehorn model. Its presence greatly affect the performance of the algorithms. The consequences of the sampling error will be discussed in the next chapter.

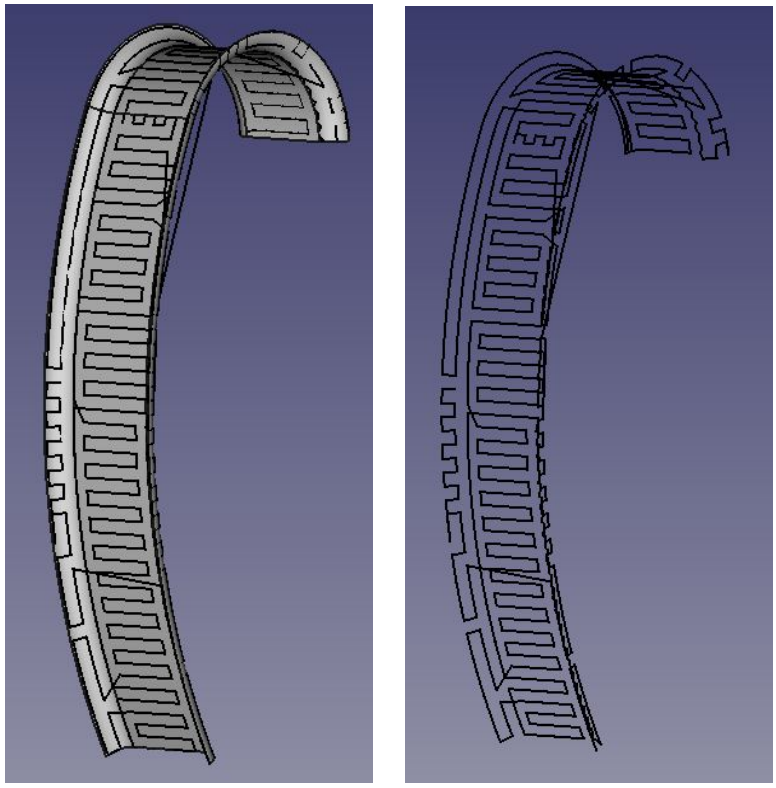
5.2.1 Greedy choice algorithm

The first path algorithm to be implemented was the greedy choice algorithm. To visualize the path, a wire connecting the samples was drafted.

In figure 5.8, the greedy choice algorithm was tested on a point cloud consisting of all faces of the shoehorn model. It can easily be seen that the path is far from optimal. There are several instances where the path intersects itself which was one of the main things that should be avoided as discussed in section 4.4.1. Furthermore, the path has some jumps from one sample to another which can be seen especially in figure 5.8b on the backside of the shoehorn. Unless it is possible to turn off the material flow during the jump, this will cause large errors between the original CAD model and the print.

In figure 5.9, the greedy choice algorithm is tested on the propeller blade. Like for the shoehorn model, the path here is far from optimal.

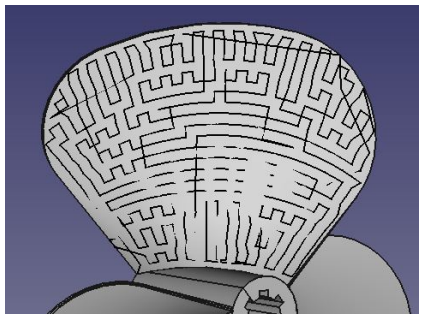
To further test the quality of the greedy path algorithm without the presence of the sampling error, it was decided to test the algorithm on one face at a time. In figure 5.10a, the algorithm is tested on the largest face of the shoehorn. Here it can be seen



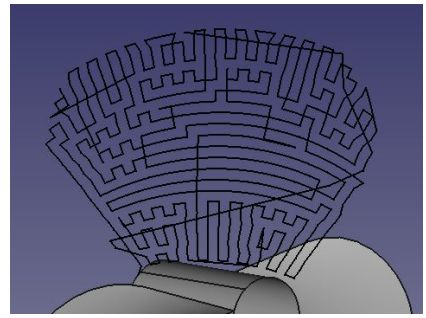
(a) Path generated using greedy algorithm on shoehorn model - with CAD model

(b) Path generated using greedy algorithm on shoehorn model - only path

Figure 5.8: Path generated using greedy algorithm on shoehorn model



(a) Path generated using greedy algorithm on propeller blade - with CAD



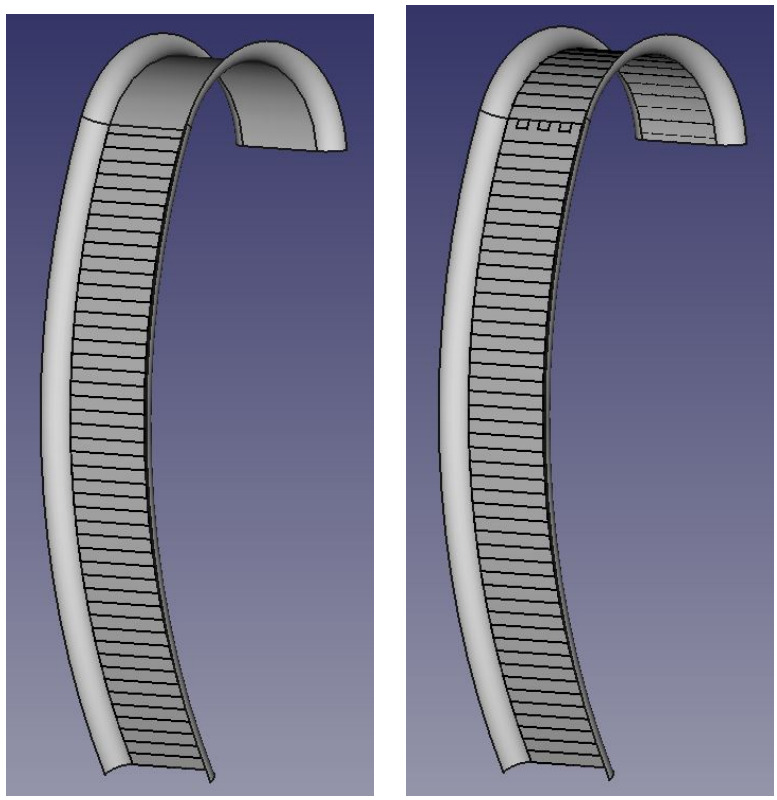
(b) Path generated using greedy algorithm on propeller blade - only path

Figure 5.9: Path generated using greedy algorithm on propeller blade

that the greedy path algorithm generates a path without self-intersections or any other problems. In figure 5.10b, the algorithm is tested on the two largest faces in the shoehorn model. This means including the face with the overhang feature. It can be seen that the path is without self-intersections and overall a good path. The path is, however, not optimal in the seam between the two faces. This is explained by the fact that the distance between the samples in the seam is not the same as for the other samples due to the sampling error.

5.2.2 Weighted greedy algorithm

The second algorithm that was tested was the weighted greedy algorithm. Initially, the weighting was based on u. That means that if the next sample had another u-value than the current u-value, a weighting value was added to the distance calculation. This was to ensure that the path followed the current u-value, that is the current level, before moving up along the model. The second weighting to be tested was y-weighting. Here, a weight is added on the y-component of the distance calculation. This was to ensure that the path followed the current y-level before moving up along the model. y-weighting was only tested on the shoehorn model since the coordinate system of the propeller blade model was different and thus not applicable. The weight was set to



(a) Path generated using greedy algorithm on largest face of shoehorn model

(b) Path generated using greedy algorithm on the two largest faces of shoehorn model

Figure 5.10: Path generated using greedy algorithm on parts shoehorn model

10.0 for both weightings, the double of the distance between the samples.

u-weighting

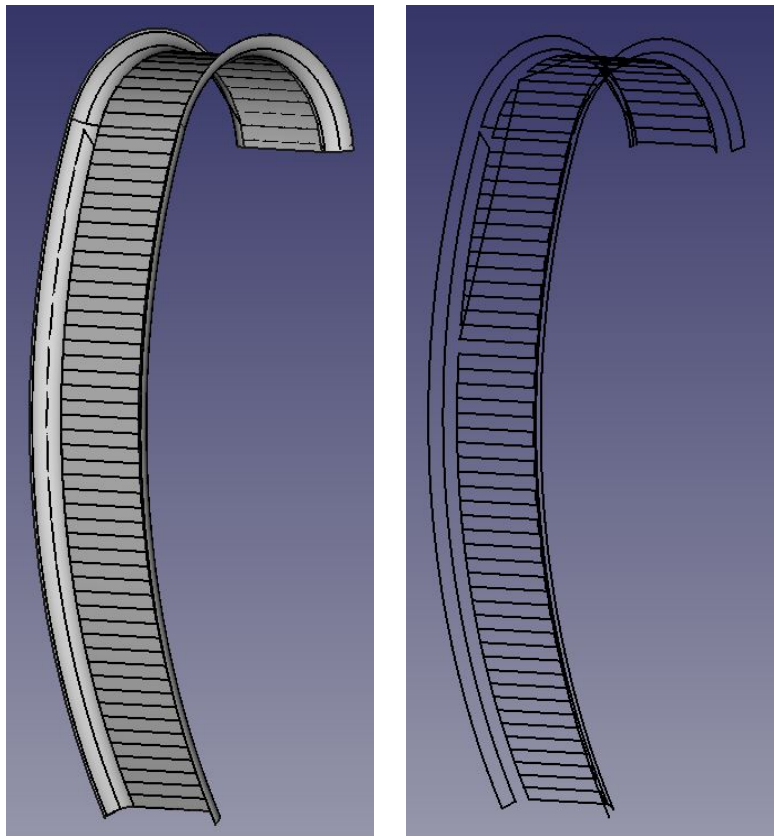
The result from the weighted greedy path algorithm tested on the shoehorn is shown in figure 5.11. It can be seen that the path completes the two largest faces first like in figure 5.10b before moving on to the sides. Compared to the result from the greedy algorithm, figure 5.8, the path is improved. The problem of self-intersections is gone, but there is still a jump on the path that is far from optimal.

The result from the weighted greedy path algorithm run on the propeller blade is shown in figure 5.12. Here it can be seen that the generated path follows the curvature of the seam between the propeller blade and the center cylinder and is without any self-intersections or jumps. It is greatly improved compared to the result from the greedy algorithm from figure 5.9.

One big problem with u-weighting on the shoehorn model is that the u and v directions are defined orthogonal on each other on the two largest faces and the sides. To take advantage of this, one solution is multi-directional slicing similar to the one presented in section 2.3.2. This means first printing the two large faces, including the overhang before printing one side at the time onto the print. A path for such a solution is presented in figure 5.13. Here, the path for the two largest faces is generated first before the path for the sides is generated afterwards, one at a time. The path generated for the two largest faces is presented in black while the path for the sides is in green and purple. By examining the figures, it can be seen that the three paths are without any self-intersections or jumps that has been recurring issues before. Based on the assumptions made about fast-curing material, this is a path that should be feasible.

y-weighting

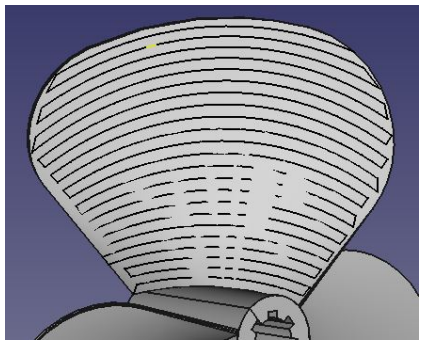
The other weighting that was tested was y-weighting. y-weighting means that a weight was added on the y-component in the distance calculation. The reason why y-weighting was implemented was that the coordinate system for the shoehorn model was defined with y up along the shoehorn. The goal was to overcome the issue with u



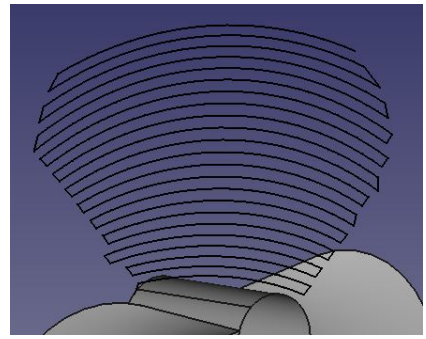
(a) Path generated using greedy algorithm with u-weighting on shoehorn model - with CAD

(b) Path generated using greedy algorithm with u-weighting on shoehorn model - only path

Figure 5.11: Path generated using greedy algorithm with u-weighting on shoehorn model



(a) Path generated using greedy algorithm with u-weighting on propeller blade - with CAD

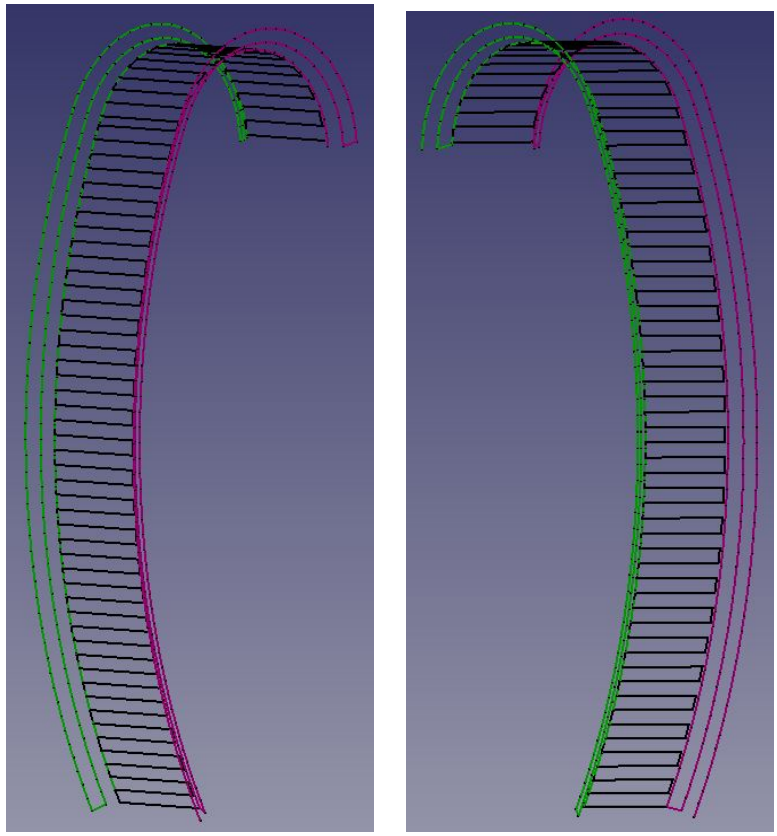


(b) Path generated using greedy algorithm with u-weighting on propeller blade - only path

Figure 5.12: Path generated using greedy algorithm with u-weighting on propeller blade

and v being defined in different directions for the two large faces and the two sides.

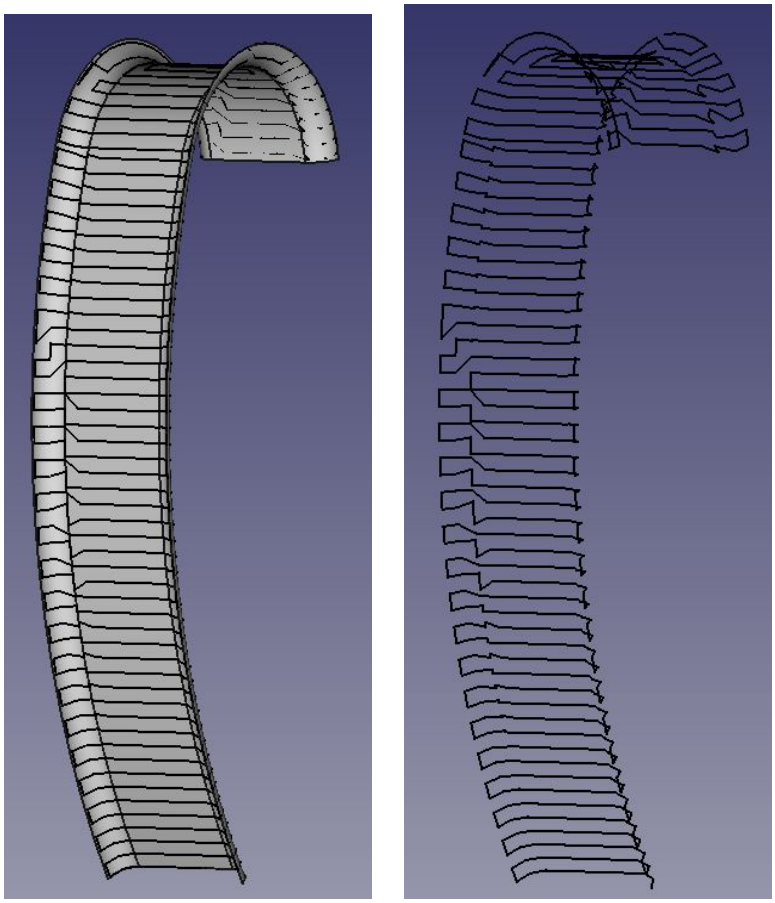
The result from the test with y-weighting is shown in figure 5.14. It can be seen that in the bottom of the model, the path moves from right to left, following the current level before moving upwards. However, as the path moves upwards, the path start having trouble. This can be explained by the error due to sampling as shown in figure 5.5. Further up along the model, the error becomes less as the sampling error become less significant. At the top of the model, a new problem arise. This can be seen in figure 5.15. The path no longer moves from left to right as it did further down on the model. This is due to the fact that y-weighting is not effective when the model bends. The movements in y is then small while the movements in the $x - z$ plane is large. As a result, the resulting algorithm acts as a greedy choice algorithm until the movements in the y-direction becomes more significant.



(a) Path generated using greedy algorithm with u-weighting on sub objects of shoehorn model - from right

(b) Path generated using greedy algorithm with u-weighting on sub objects of shoehorn model - from left

Figure 5.13: Path generated using greedy algorithm with u-weighting on sub objects of shoehorn model



(a) Path generated using greedy algorithm with y-weighting on shoehorn model - with CAD

(b) Path generated using greedy algorithm with y-weighting on shoehorn model - only path

Figure 5.14: Path generated using greedy algorithm with y-weighting on shoehorn model

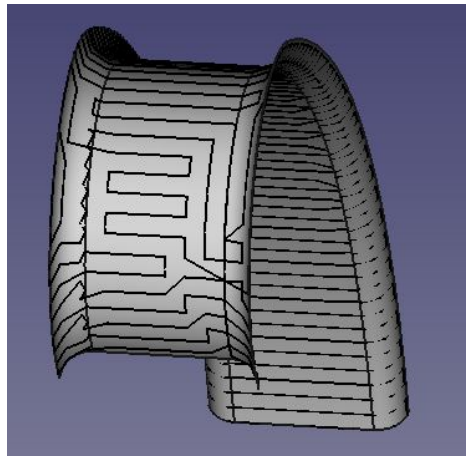


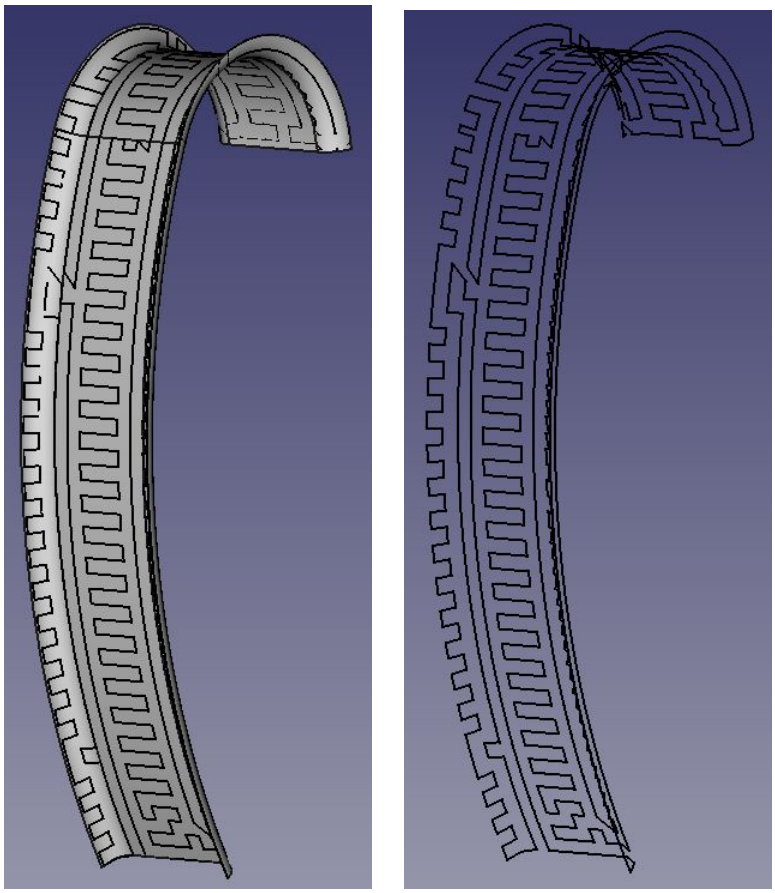
Figure 5.15: Path generated using greedy algorithm with y-weighting on shoehorn model - from top

5.2.3 Travelling Salesman Problem algorithm

The last algorithm to be implemented was the Travelling Salesman Problem (TSP) algorithm. This was done using the modified TSP solver as described in section 4.4.3. The result from running the TSP algorithm on the shoehorn model is shown in figure 5.16. It took several minutes to generate the path. Even though the path does not have any self-intersections or jumps, it is far from optimal.

The result from running the TSP algorithm on the two largest faces is shown in figure 5.17. This path took a little more than two minutes to generate. In figure 5.17b it can easily be seen why it is an issue that the salesman wants to return to the same city. The path that goes up along the left side of the model is a direct result of this assumption.

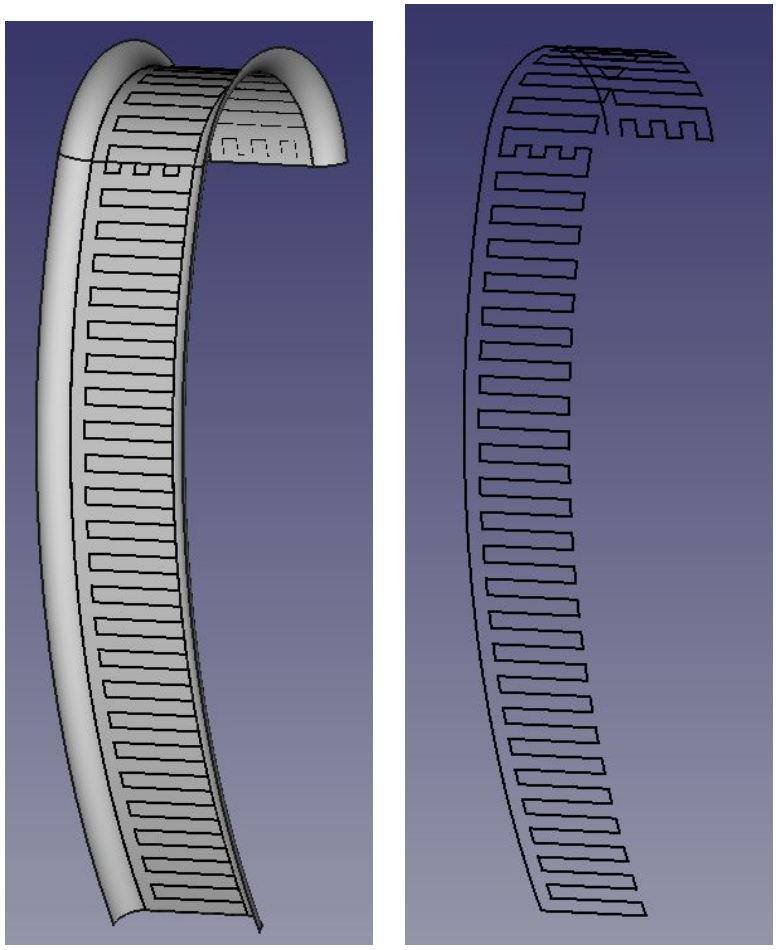
The result from running the TSP algorithm on the propeller blade is shown in figure 5.18. Like with the path generated for the shoehorn using TSP, it is far from optimal and took a long time to generate.



(a) Path generated using TSP on shoehorn model - with CAD

(b) Path generated using TSP on shoehorn model - only path

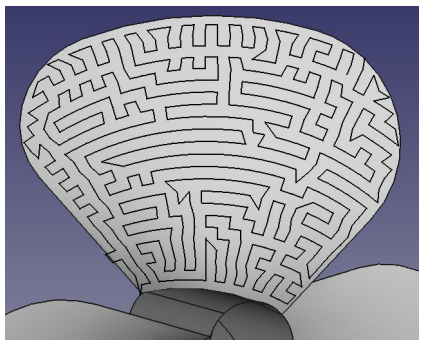
Figure 5.16: Path generated using TSP on shoehorn model



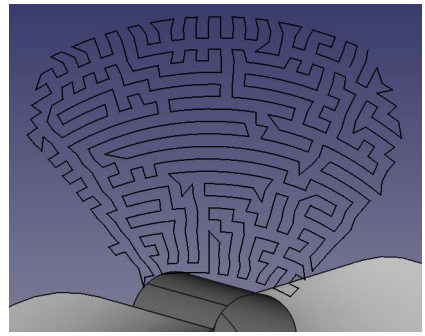
(a) Path generated using TSP on the two largest faces of shoehorn model - with CAD

(b) Path generated using TSP on the two largest faces of shoehorn model - only path

Figure 5.17: Path generated using TSP on the two largest faces of shoehorn model



(a) Path generated using TSP on propeller blade - with CAD



(b) Path generated using TSP on propeller blade - only path

Figure 5.18: Path generated using TSP on propeller blade

Chapter 6

Discussion

In chapter 3, the specialization project including some improvements was presented and discussed. Based on this, a new solution was presented in chapter 4. The results from the new method is presented in chapter 5. In this chapter, those results will be reviewed more closely and discussed. At the end of the chapter, the system as a whole will be discussed.

6.1 Sampling

The first step of the AM path generation system was to sample the surface of a CAD model. This is realized through sampling each sub object, face, separately before gathering all the samples into a joined point cloud. The advantages of this method is that the Sample object can be customized to contain the desired information. Furthermore, it is easy to generate a path based on this point cloud. There is, however, one main disadvantage of this method. That is lost curvature of the model due to the sampling. The sampling is a form of discretization meaning that the continuous surface is transferred into a discrete counterpart. To minimize the error between the CAD model and the point cloud, the step length can be reduced. This will, on the other hand cause a larger point cloud which again will make the path generation system slower.

As a result, there is a trade off between discretization error and the effectiveness of the system.

In section 5.1, the results of the sampling of the shoehorn model and the propeller blade is presented. By looking at the figures in the section, it can be seen that the sampling method managed to sample the surfaces and captures most of the curvature of the surfaces. On the shoehorn model, the distance between each sample was good without calibration in each iteration. For the sampling of the propeller blade, however, the distance between each sample was not the same over the whole surface. At the bottom of the blade, in the seam between the blade and the center, the distance between each sample was small while large at the top of the blade. This can be seen in figure 5.6. The problem was fixed by running the calibration of the step length between each sample. This resulted in an even distance between each sample and the result can be seen in figure 5.7.

One great advantage of sampling along the surface as in this method using the parameter coordinates u and v, is that the resulting samples follows the curvature of the surface. This can be seen in figure 5.7. The samples follow the curvature of the seam between the propeller blade and the center. This makes it easy to generate a path that follows the same curvature.

Even though the distance became the same between each sample as wanted, it can be discussed if that is the best solution. For surfaces with high curvature, it can be desirable to introduce adaptive step length in other to capture the curvature of the surface better. This can for example be realized by comparing the normal vectors of two samples with each other. If the angle between them is larger than some limit, the step length is reduced and the current sample is deleted to find a new one. It works the same the other way around, if the angle between the normal vector is less than some limit, the step length is increased. In this way, the number of samples in high curvature areas is increased while reduced on flat surfaces. All in all, adaptive step length does not have to mean more samples in the point cloud in the end.

The biggest problem with the sampling method is the issue with sampling between different faces. A figure showing this issue was presented in figure 5.5. This issue becomes especially large when u and v are defined in different directions for different

faces as in the shoehorn model. The presence of the sampling error can be seen in two ways. The first is on the edge between faces. Edges are sampled twice, once for each face. This causes twice as many samples as supposed to along the edges. A solution to this problem would be to remove the samples that are within some radius of each other. The sampling error is visible in another way as well. In figure 5.5, it can be seen that there is an offset between the samples of the two largest faces and the sides of the shoehorn model. This offset grows larger up towards the middle of the shoehorn before it decreases again. The offset is caused by the fact that the parameter coordinates of the faces are defined in different directions. Unfortunately, this is not an error that can be fixed easily. One solution might be to rotate the coordinate systems so that all faces are defined in the same direction, or to sample the faces without using the parameter coordinates. The sampling issue as it is greatly affects the path generation algorithms as will be discussed in the following section.

One goal with the new method was to avoid the STL format since it is an approximation of the surface. The continuous format STEP was used instead of STL, but when sampled, the result is still an approximation of the surface. It can therefore be argued that the STL format could be used. This is the format that is used in most applications within AM. There are, however, some consequences of using STL instead of STEP. A 3D model that is represented using STL has the whole surface approximated as one using polygons. This means that the sampling problem between faces using the method presented in this thesis would no longer be a problem. The downside is that since the whole surface is represented as one, and not sub surfaces, it is more difficult to sample and print just some parts of the model if desirable. For the same reason, multi-directional sampling and printing becomes more difficult.

6.2 Path generation

6.2.1 Greedy choice algorithm

The first algorithm to be tested was the greedy choice algorithm. The result from running this algorithm on a point cloud of the entire shoehorn model and on the

propeller blade is shown in figure 5.8 and 5.9, respectively. It can easily be seen that neither of those paths are optimal for AM. It is difficult to say just how much the algorithm is affected by the sampling error. To get a better idea, a path was generated for just that face as well as for the two largest faces since the path for the shoehorn model had some good tendencies on the two largest faces. The results can be seen in figure 5.10. Here it can be seen quite clearly that the paths are improved and has become feasible AM paths. In the seam between the two faces in figure 5.10b, there are still some trouble in the path, however.

The greedy choice algorithm is based on the assumption that making a locally optimal choice will lead to a global optimal solution. A question to ask here is: what is a local optimal choice in the sense of AM path generation? In the implementation of the algorithm, the locally optimal choice is set to be the closest sample with regards to distance. If the sampling had been optimal, meaning the same distance between all points, this might have been true. This is the case that is seen when the path is generated for only the largest faces of the shoehorn model. When more than one face is introduced, the sampling is no longer optimal and the algorithm falls through. For the propeller blade, the algorithm falls through due to the curvature of the blade which makes the distance between every sample uneven. However, even though perfect sampling would have improved the outcome of the greedy algorithm, there is no guarantee that it will print in one direction to the end before moving on to the next level.

6.2.2 Weighted greedy choice algorithm

To better control the printing direction, the weighted greedy algorithm was implemented. There were two different weightings implemented: u-weighting and y-weighting. The results from testing the algorithm with the two weightings can be seen in section 5.2.2.

u-weighting

The first weighting to be tested was the u-weighting. u-weighting means adding a weight on the distance calculation in the weighted greedy algorithm if the u-value is different from the u-value of the current sample. This was to ensure that the path followed the current u-value before moving on to the next. The result from testing the u-weighting can be seen in figure 5.11 and 5.12. In figure 5.11, the path is generated for the shoehorn model. It can be seen that the path follows the current u-value as supposed to. The only problem is that u and v is, like mentioned earlier, defined orthogonal on each other on the two largest faces and the sides. As a result, the path has a jump that can be seen in figure 5.11b making this path not optimal.

In figure 5.12, the algorithm is tested on the propeller blade. Here it can be seen that the path is quite satisfactory. It follows the curvature of the seam between the propeller blade and the center cylinder and completes one direction before moving on to the next level. This is a path that is infeasible using a traditional 3D printer and thus not possible to generate using traditional AM software. None of the other existing solutions presented in 2.1.3 has generated a path resembling this one either. Printing a propeller blade using this path would save material compared to using traditional machining.

As mentioned, the problem with the shoehorn model is that the parameter coordinates u and v are defined in different directions for the different faces. To overcome this issue, it was decided to run the greedy algorithm with u-weighting on the faces that were defined differently one at a time. This means the two largest faces together and the the two sides separately. The result can be seen in figure 5.13. Three different paths was generated and they are represented in green, black and purple. By looking at the three paths, it can be seen that they are all satisfactory with no issues like self-intersections or jumps. However, this path would have to be printed in three runs and this does not come without complications. Most likely, the two largest faces would be printed first, the black path, before printing the sides onto the print. This resembles multi-directional slicing as presented in section 2.3.2. The downside of this method is that AM has great uncertainties, especially in large scale. It is very likely that there

will be some difference between the print and the 3D model. So when the two largest faces are printed first, there will some differences. This becomes a problem when the path of the two sides are based on the 3D model of the shoehorn and not the print. Those differences can cause even larger differences in the print of the sides or even worse, cause collision between the robot and the print. To overcome the last problem, reactive collision avoidance could be implemented in the robot control system. The first problem could be solved by having an online path-planning system that can alter the generated path based on the print.

The paths generated in figure 5.13 are based on the assumption of fast curing material making it possible to print in any direction without support structures. Stratasys, as presented in section 2.1.3, has found a method to overcome this problem. They made a system with 8 DOF that rotates the part for optimal printing direction. In this way, the area to be printed is always pointed upwards meaning that overhang without support structures is no problem. The shoehorn model could be printed in the same way by rotating the part between each path. For example could the black path be printed first, then turn the part onto one side to print the side and then turn again to print the other side. To realize this, a method to turn and stabilize the printed part would be necessary. Furthermore, it would require a system that could see the print and be able to know where the path should be printed onto it.

y-weighting

The second weighting to be tested was y-weighting. y-weighting means adding a weight on the y-component of the distance calculations in the weighted greedy algorithm. The reason y-weighting was implemented was that the coordinate system of the shoehorn model is defined with y up along the model. By adding a weight on y, the path would complete one layer in the $x - z$ plane before moving onto the next. The hope was that y-weighting would overcome the problem of u and v being defined in different directions for the different faces of the shoehorn model. The results using the y-weighting algorithm is given in figure 5.14 and 5.15. It can be seen that the path starts out good moving from right to left before moving on to the next level. The problem starts half way up along the model when the sampling error is growing larger.

Also, at the top of the model where the desired printing direction is no longer y, the path is far from optimal as can be seen in figure 5.15. Despite the mentioned problems, this is a weighting system that shows promise. If the sampling had been without errors, it is likely that the generated path would have been greatly improved. Out of the tested algorithms, this is the only one that generate a path that move from right to left, completing one level before moving on to the next for the shoehorn model. It is important to mention that the path issue at the top of the shoehorn would not have been improved with improved sampling.

In this thesis it was only y-weighting that was implemented and tested, but x and z weighting could just as easily have been implemented. The drawback with all of these is that they only work if the printing direction of the model moves in the direction of the weighting. For the propeller blade, there is no constant building direction in $x - y - z$ and this type of weighting is therefore not applicable. For the shoehorn, it worked fine, besides the sampling error, until the top of the shoehorn where the printing direction change.

6.2.3 TSP algorithm

The last algorithm to be tested was TSP. The result from testing the algorithm can be seen in figure 5.16, 5.17 and 5.18. By looking at figure 5.16, it can be seen that the path does not have any of the previously mentioned problems such as self-intersections or jumps, but the path is still far from optimal. To see how much of the problem is caused by the sampling error, TSP was tested on the two largest faces as well as shown in figure 5.17. Even though the path is greatly improved, it is still not optimal. One founding assumption in TSP is that the salesman want to start in a city and return to the same city in the end. This is an assumption that turned out to make the TSP algorithm less applicable to the AM path generation problem than expected. It can be seen that the assumption that the salesman wants to return to the starting city is an issue in figure 5.17b. The path moves up along the left part of the model. This is a direct cause of that assumption.

The fact that TSP is NP-complete causes another major issue with the TSP algorithm.

When the point cloud grows large, it takes a lot of time to generate a path. Generating the path for the entire shoehorn model took several minutes. Here, the distance between each sample was 10.0 mm x 10.0 mm. The distance could have been much less causing a much larger point cloud. This makes TSP very inefficient and not very applicable since an efficient algorithm was one of the desired qualities.

TSP was presented in section 2.5.3 as a special case of an Hamiltonian cycle where the goal is to find the shortest path. In the core of both, is the assumption of returning to the home city after making a tour. To avoid this basic assumption, one trick can be made. This is presented in Lawler et al. (1985) as the computer wiring problem. The book suggests to create a dummy point whose distances to every point is 0. After finding a path, the dummy point is deleted and the result is a TSP path that does not return back to the home city. By implementing this modified TSP algorithm, the resulting path might have been improved. Due to the inefficiency mentioned in the previous paragraph of TSP, this modified TSP algorithm was not considered applicable and thus not implemented and tested.

6.2.4 Comparing the algorithms

Three different algorithms has been presented, tested and discussed. The greedy choice algorithm is quick, but the path is full of self-intersections and jumps making is far from optimal for AM. The TSP algorithm is inefficient, and with the assumption of returning to the starting city, not optimal. If the mentioned solution to the computer wiring problem had been implemented, the resulting path might have been improved, but the algorithm would still be very slow. Out of the three algorithms, weighted greedy is the one that showed the most promise. Two different types of weighting was tested and discussed, u-weighting and y-weighting. The two weightings are good at different thing and there is not one solution that is better for all purposes.

For the propeller blade, the best algorithm is without a doubt greedy algorithm with u-weighting. With that algorithm the generated path followed the curvature of the seam between the blade and the center cylinder. The path completed one level before moving on to the next and the path was without any issues such as self-intersections

and jumps. This is a path that takes advantage of the DOF of the robot manipulator through following the curvature of the propeller blade and it is therefore infeasible using traditional solutions for AM.

For the shoehorn model, there was not one clear answer to what the best algorithm was. With the u-weighting, the path generated for the shoehorn as a whole was far from optimal. However, if the path was generated in three parts, the two largest faces together and each side separately, the path was greatly improved. The downside of this method is that the part has to be printed in three separate runs which causes new problems as discussed above. With the y-weighting, the path moved from right to left, one level at a time as supposed to up until the sampling error became too significant. This is the only method that managed to generate a path moving from one side of the model to the other, printing the sides in the same level as the largest face in the middle of the model. The downside with y-weighting is that it only works as long as the printing direction is in the y-direction, which was not the case at the top of the shoehorn model.

In section 6.1, the sampling error was discussed, and as mentioned there, it became a big problem for all the path generation algorithms. It is very difficult to generate an optimal path when the point cloud the algorithms use is far from optimal. The algorithms that worked the best here was the ones that was best on handling the sampling error. It is hard to say how large an impact an optimal point would have given.

6.3 Path generation system

In chapter 3, the specialization project with some further work was presented. Here it was concluded that it was necessary with a new and improved method to better control the process and to take advantage of the DOF. As a result, a new method has been developed, tested and discussed. In the new method, the created python script controls the entire process. This is an improvement compared to the old method where the process was considered to be difficult to control. Another improvement with the new method is that it takes better advantage of the DOF. The path generated using

the greedy algorithm with u-weighting on the propeller blade follows the curvature of the blade. This would not have been possible with a 3 DOF 3D printer and thus not been possible to generate using the old method. It is however important to mention that the solution in the specialization project was a system that took in a CAD model and returned RAPID code simulated and verified using the powerful virtual controller in RobotStudio. The new method takes in a CAD model in the same way, but only generates a path. The tool-path planning problem is considered to be very complex and has to be solved before the generated path is ready to be tested on a physical robot.

Like mentioned, the method presented in this thesis only generates a path, not tool-path or robot code. There are other limitations to the solution as well. Two assumptions that were made was that the method is made for simple surfaces and require fast-curing material that enables printing in mid-air. As a result, solid models are thus not possible to print using this method. Also, printing complex surfaces can challenge the reachability of the robot. To enable printing such complex surfaces, the path might have to be planned in a more strategic way. There are not a lot of materials that enable mid-air printing. All traditional 3D printing material require support structures for the print to hold. To enable printing for example overhang, the path might have to be altered to be feasible or the part has to be rotated. Another limitation of the presented solution is that it is only tested on two different models, the shoehorn model and the propeller blade. To further guarantee its behaviour, the solution has to be tested on more models.

Even though the sampling results is an approximation of the surface, this does not mean that the end result would be any different with a continuous path. Large-scale AM has large inaccuracies. If the step length of the sampling is short enough, it is likely that the end-result would be the same as if the path was continuous. Furthermore, a discrete path can be made continuous through robot systems. In ROS, which was presented in section 2.4, a path can be represented as a series of way points. In MoveIt!, Cartesian Paths given as a series of way points can be interpolated with a desired resolution. The result is a discrete path made continuous.

Chapter 7

Conclusion

In this thesis, a method for generating a path for AM using a 6 DOF robot manipulator for extrusion based AM based on simple surfaces of a CAD model was presented. The method is implemented using python and ran as a macro in FreeCAD. First, the surface(s) is sampled with a desired step length before a path algorithm is ran on the generated point cloud. Three different algorithms was implemented and tested: greedy choice, weighted greedy choice and TSP. Greedy choice generated a path effectively, but it was full of self-intersections. TSP was very slow and as implemented with the assumption of returning to the starting sample, not applicable. Out of the three, weighted greedy was the algorithm that gave the best result. With u-weighting, the generated path completed one u-level before moving on to the next. The path generated for the propeller blade followed the curvature of the blade as can be seen in figure 5.12. This is a path that would not have been feasible with a traditional AM solutions. For the shoehorn model, there were some issues with the different faces being defined differently with respect to the parameter coordinates u and v. One solution to this was generating a path for one surface at a time resembling multi-directional slicing as shown in figure 5.13. The path generated for the shoehorn model enables printing in overhang without the need for support structures. This is only feasible under the assumption of fast-curing material enabling mid-air printing. With y-weighting in

the weighting algorithm, the path for the shoehorn model moved from right to left before moving on to the next level. The path started to struggle further up along the shoehorn when the sampling error became too significant as can be seen in figure 5.14.

In summary, the developed method generates paths for AM that takes advantage of the 6 DOF of the robot manipulator. The method is effective and by interfacing with FreeCAD, it is easy to review the generated path through visual aids.

7.1 Prospect for future work

- Improve the sampling method. The sampling error between faces caused problems for the path generation algorithms. To further improve the paths, the sampling would have to be improved first. Furthermore, an adaptive step length could be implemented to better catch the structure of curved surfaces.
- Develop new algorithms for simple surfaces to improve the result. For the shoehorn model, there was not one algorithm that gave an optimal path even though both u and y weighting showed promise.
- Develop a graphical user interface (GUI) that makes it easier to choose accuracy of the sampling and to choose between the different algorithms.
- Develop new algorithms that generates paths for more complex geometries such as solids.
- Develop a method for generating a tool-path based on the generated paths using the method in this thesis. This is a complex problem, but necessary to realize the generated paths.

References

17 best 3D slicer software tools for 3D printers (2017). https://all3dp.com/1/best-3d-slicer-software-3d-printer/. [Online; accessed 06-December-2017].

ABB (2007). *Introduction to RAPID*.

ABB Robotics (2017). http://new.abb.com/products/robotics. [Online; accessed 17-December-2017].

Allen, R. J. and Trask, R. S. (2015). An experimental demonstration of effective curved layer fused filament fabrication utilising a parallel deposition robot, *Additive Manufacturing* **8**: 78–87.

Alsharhan, A. T., Centea, T. and Gupta, S. K. (2017). Enhancing mechanical properties of thin-walled structures using non-planar extrusion based additive manufacturing, *ASME 2017 12th International Manufacturing Science and Engineering Conference collocated with the JSME/ASME 2017 6th International Conference on Materials and Processing*, American Society of Mechanical Engineers, pp. V002T01A016–V002T01A016.

AM basics (2017). http://additivemanufacturing.com/basics/. [Online; accessed 16-December-2017].

Brogårdh, T. (2007). Present and future robot control development—an industrial perspective, *Annual Reviews in Control* **31**(1): 69–79.

Chua, C. K., Leong, K. F. and Lim, C. S. (2003). *Rapid prototyping: principles and applications*, Vol. 1, World Scientific.

Cormen, T. H. (2009). *Introduction to algorithms*, MIT press.

CraftWare (2017). `https://craftunique.com/craftware/`. [Online; accessed 17-November-2017].

Ding, D., Pan, Z., Cuiuri, D., Li, H., Larkin, N. and Van Duin, S. (2016). Automatic multi-direction slicing algorithms for wire based additive manufacturing, *Robotics and Computer-Integrated Manufacturing* **37**: 139–150.

Ding, D., Pan, Z. S., Cuiuri, D. and Li, H. (2014). A tool-path generation strategy for wire and arc additive manufacturing, *The international journal of advanced manufacturing technology* **73**(1-4): 173–183.

Evans, B. (2012). *Practical 3D printers: The science and art of 3D printing*, Apress.

Evjemo, L., Moe, S., Gravdahl, J., Roulet-Dubonnet, O., Gellein, L. and Brøtan, V. (2017). Additive manufacturing by robot manipulator: An overview of the state-of-the-art and proof-of-concept results, *In proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus*.

FreeCAD (2018). `https://www.freecadweb.org/`. [Online; accessed 15-March-2018].

G Code (2017). `http://gcodes.net/`. [Online; accessed 06-December-2017].

Gibson, I., Rosen, D. W., Stucker, B. et al. (2010). *Additive manufacturing technologies*, Vol. 238, Springer.

Grasshopper (2017). `http://www.grasshopper3d.com/`. [Online; accessed 17-December-2017].

HAL Robotics (2017). `http://hal-robotics.com/`. [Online; accessed 17-December-2017].

Hoye, N. (2015). Characterisation of ti-6al-4v deposits produced by arc-wire based additive manufacture, *University of Wollongong Thesis Collections* . A thesis submitted in partial fulfillment of the requirements for the award of the degree Doctor of Philosophy.

Huang, B. and Singamneni, S. (2012). Alternate slicing and deposition strategies for fused deposition modelling of light curved parts, *J. of Achievem in mat and manuf* **55**(2): 511–517.

Jamieson, R. and Hacker, H. (1995). Direct slicing of cad models for rapid prototyping, *Rapid Prototyping Journal* **1**(2): 4–12.

Joris Laarman Lab (2018). `http://www.jorislaarman.com/`. [Online; accessed 07-May-2018].

Kirschman, C. and Jara-Almonte, C. (1992). A parallel slicing algorithm for solid freeform fabrication processes, *Solid Freeform Fabrication Proceedings, Austin, TX* pp. 26–33.

Kubo, M. and Pedroso, J. (2009). Metaheuristics: A programming guide, *Kyoritsu Shuppan Co., Ltd., Tokyo* .

KUKA (2017). `https://www.kuka.com/`. [Online; accessed 17-December-2017].

Lasemi, A., Xue, D. and Gu, P. (2010). Recent development in cnc machining of freeform surfaces: A state-of-the-art review, *Computer-Aided Design* **42**(7): 641–654.

Lawler, E. L., Lenstra, J. K., Kan, A. R., Shmoys, D. B. et al. (1985). *The traveling salesman problem: a guided tour of combinatorial optimization*, Vol. 3, Wiley New York.

Lim, S., Buswell, R. A., Valentine, P. J., Piker, D., Austin, S. A. and De Kestelier, X. (2016). Modelling curved-layered printing paths for fabricating large-scale construction components, *Additive Manufacturing* **12**: 216–230.

Lin, Z., Fu, J., Shen, H., Gan, W. and Yue, S. (2015). Tool path generation for multi-axis freeform surface finishing with the lkh tsp solver, *Computer-Aided Design* **69**: 51–61.

Livesu, M., Ellero, S., Martínez, J., Lefebvre, S. and Attene, M. (2017). From 3d models to 3d prints: an overview of the processing pipeline, *Computer Graphics Forum*, Vol. 36, Wiley Online Library, pp. 537–564.

Machining (2017). http://fab.cba.mit.edu/classes/863.12/people/laia.mogassoldevila/projects/p7.html. [Online; accessed 16-December-2017].

Mataerial (2018). http://mataerial.com/. [Online; accessed 07-May-2018].

Metal Finishing Industry (1995). http://infohouse.p2ric.org/ref/03/02454/overview.htm. [Online; accessed 16-April-2018].

MoveIt! (2018). http://moveit.ros.org/. [Online; accessed 15-March-2018].

MX3D (2018). http://mx3d.com/. [Online; accessed 11-April-2018].

New Story + ICON: 3D printed houses (2018). https://www.iconbuild.com/home. [Online; accessed 11-April-2018].

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*, Springer.

Pan, Z., Ding, D., Wu, B., Cuiuri, D., Li, H. and Norrish, J. (2018). Arc welding processes for additive manufacturing: A review, *Transactions on Intelligent Welding Manufacturing*, Springer, pp. 3–24.

Pan, Z., Polden, J., Larkin, N., Van Duin, S. and Norrish, J. (2012). Recent progress on programming methods for industrial robots, *Robotics and Computer-Integrated Manufacturing* **28**(2): 87–94.

Robotic composite 3D demonstrator (2017). http://blog.stratasys.com/2016/08/24/infinite-build-robotic-composite-3d-demonstrator/. [Online; accessed 16-December-2017].

Robotmaster CAD/CAM (2017). http://www.robotmaster.com/en/. [Online; accessed 17-December-2017].

RobotStudio® (2017). http://new.abb.com/products/robotics/robotstudio. [Online; accessed 06-December-2017].

RobotStudio Machining PowerPac (2017). http://new.abb.com/products/robotics/application-software/machining/robotstudio-machining-powerpac. [Online; accessed 05-December-2017].

Rogers, D. F. (2000). *An introduction to NURBS: with historical perspective*, Elsevier.

ROS (2018). http://www.ros.org/. [Online; accessed 15-March-2018].

Sabourin, E., Houser, S. A. and Helge Bøhn, J. (1996). Adaptive slicing using stepwise uniform refinement, *Rapid Prototyping Journal* **2**(4): 20–26.

Sarcar, M., Rao, K. M. and Narayan, K. L. (2008). *Computer aided design and manufacturing*, PHI Learning Pvt. Ltd.

SelfCAD (2018). https://www.selfcad.com/. [Online; accessed 26-January-2018].

Smid, P. (2003). *CNC programming handbook: a comprehensive guide to practical CNC programming*, Industrial Press Inc.

Stratasys (2017). http://www.stratasys.com/. [Online; accessed 28-October-2017].

Stratasys demonstrates next generation 3D printing technology for large parts composites (2016). https://www.additivemanufacturing.media/news/. [Online; accessed 18-October-2017].

Stroud, I. (2006). *Boundary representation modelling techniques*, Springer Science & Business Media.

Taufik, M. and Jain, P. K. (2016). A study of build edge profile for prediction of surface roughness in fused deposition modeling, *Journal of Manufacturing Science and Engineering* **138**(6): 061002.

The construction of Europe's first 3D printed building has begun (2017). https://3dprinthuset.dk/europes-first-3d-printed-building/. [Online; accessed 16-December-2017].

The STEP Standard (2018). https://www.steptools.com/stds/step/step_1.html. [Online; accessed 07-February-2018].

TSP solver (2018). http://www.dcc.fc.up.pt/~jpp/code/py_metaheur/tsp.py. [Online; accessed 20-May-2018].

Wah, P. K., Murty, K. G., Joneja, A. and Chiu, L. C. (2002). Tool path optimization in layered manufacturing, *Iie Transactions* **34**(4): 335–347.

Yang, P., Li, K. and Qian, X. (2011). Topologically enhanced slicing of mls surfaces, *Journal of Computing and Information Science in Engineering* **11**(3): 031003.

Zhang, G. Q., Mondesir, W., Martinez, C., Li, X., Fuhlbrigge, T. A. and Bheda, H. (2015). Robotic additive manufacturing along curved surface—a step towards free-form fabrication, *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, IEEE, pp. 721–726.

Zhang, G. Q., Spaak, A., Martinez, C., Lasko, D. T., Zhang, B. and Fuhlbrigge, T. A. (2016). Robotic additive manufacturing process simulation-towards design and analysis with building parameter in consideration, *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, IEEE, pp. 609–613.

Zheng, H., Cong, M., Dong, H., Liu, Y. and Liu, D. (2017). Cad-based automatic path generation and optimization for laser cladding robot in additive manufacturing, *The International Journal of Advanced Manufacturing Technology* **92**(9-12): 3605–3614.