

Helene Boge

Robotic additive manufacturing: Testing and evaluation of curved layer generation

Master's thesis in Cybernetics and Robotics

Supervisor: Jan Tommy Gravdahl, Co-supervisor: Linn
Danielsen Evjemo

December 2019

Helene Boge

Robotic additive manufacturing: Testing and evaluation of curved layer generation

Master's thesis in Cybernetics and Robotics
Supervisor: Jan Tommy Gravdahl, Co-supervisor: Linn Danielsen
Ejemo
December 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics





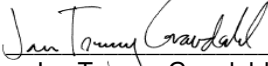
MSc thesis assignment

Name of the candidate: Helene Boge
Subject: Engineering Cybernetics
Title: Robotic additive manufacturing: testing and evaluation of curved layer generation

Background: *The topic of this thesis work is curved layer generation for industrial manipulators used for depositing material in an additive manufacturing (AM) process. Objects with geometries that are difficult, if not impossible, to manufacture in a traditional 3D printer will be studied. This typically includes objects with overhangs. This work builds on and extends the work in [1]*

Tasks:

1. Give an overview of the challenges of using additive manufacturing (AM) as a general manufacturing technology and explain how robot technology can aid in making AM a more versatile manufacturing technique.
2. Present and explain a new robotic AM system proposed in [2] for the support-free manufacturing of objects using AM.
3. Implement a greedy method for generating manufacturing layers from the robotic AM system proposed in [2] and explain why greedy methods in general may give sub-optimal results.
4. Implement an extension to the greedy method that aims to improve the greedy result, also proposed in [2].
5. Generate suitable test objects and run simulations of both programs on this group of objects. Visualize the simulation results and compare the results with each other, also compare the results to a traditional AM layer generation method.
6. Identify strengths and weaknesses in the algorithms chosen for implementation and suggest improvements to the algorithms and/or system as a whole.
7. Propose topics for further work.


Jan Tommy Gravdahl
Professor, supervisor

Co-supervisor: Linn Danielsen Evjemo

[1] H. Boge, A Literature Review on the use of Robot Technology in Additive Manufacturing, project report, Dept. of engineering cybernetics, NTNU, 2019

[2] C. Dai, C. C. L. Wang, C. Wu, S. Lefebvre, G. Fang, and Y. Liu, Support-free volume printing by multi-axis motion, ACM Transactions on Graphics, vol. 37, August 2018

Abstract

Additive manufacturing (AM), often referred to as 3D printing, is one of a total of three common manufacturing technologies of today. It is based on building a part by the accumulation of material, point by point, layer by layer. The further development of the manufacturing technology does, however, face some challenges in order for it to be a general manufacturing technique. Therefore it is today mostly used for making prototypes of early versions of parts or products. In this thesis, we investigate a new robotic AM system that has been developed in order to overcome many of the challenges associated with AM. The system has previously produced promising results in research, where it have been used to manufacture objects of complex shapes in a support-free manner. The new robotic approach to AM is based on dividing an object into curved layers and covering each layer with material in order to realize the object. In order to further identify important properties of the newly proposed system, methods for dividing objects into layers have been implemented and run on different test objects. The simulation results uncover both strengths and weaknesses in the new system. For the sake of context, the results are also compared to traditional slicing methods commercially used today.

Sammendrag

Additive manufacturing (AM), mer kjent som 3D-printing, er den siste av totalt tre teknologier brukt ved produksjon i dag. Teknologien går ut på å ”bygge” deler ved å akkumulere materiale punkt for punkt, lag for lag. Noen utfordringer relatert til denne måten å produsere deler på gjenstår å løses før teknologien kan bli ansett som en generell produksjonsteknologi. Grunnet dette, er teknologien i dag mest brukt til å lage prototyper, som er tidlige versjoner av produkter under utvikling. I denne oppgaven undersøker vi et nytt robotisk AM system som har blitt utviklet for å overkomme mange av utfordringene knyttet til AM. Systemet har tidligere blitt brukt til å produsere komplekse objekter uten støttestrukturer. Den nye robotiske tilnærmingen til AM baserer seg på å dekomponere et objekt i kurvede lag og realisere objektet ved å dekke disse lagene med materiale. For å videre identifisere viktige egenskaper ved det nye systemet, har metodene for å dekomponere objekter blitt implementert og kjørt på forskjellige test-objekter. Simuleringsresultatene avdekker både styrker og svakheter tilknyttet det nye systemet. For å sette systemet i kontekst så har resultatene også blitt sammenlignet med kommersielt populære dekomposisjons-metoder.

Preface

This thesis has been written to fulfill the graduation requirements of the MSc programme in Cybernetics and robotics at Norwegian University of Science and Technology, Autumn 2019. It is an independent work by the author, but guidance have been received in meetings with supervisors every 2-3 weeks starting from 28.08.19.

The paper in focus was discovered by the author while performing a literature review during a preliminary project of spring 2019, and all code referred to in the thesis is implemented by the author herself based on algorithms from said paper.

Acknowledgments

I would like to thank my supervisor Jan Tommy Gravdahl and co-supervisor Linn Danielsen Evjemo, both at NTNU, for valuable help and guidance throughout my work with this thesis. I would also like to thank my co-students for joining me in discussions and for helping me to look at my problems in new ways.

Contents

Problem description	i
Abstract	iii
Sammendrag	v
Preface	vii
Acknowledgments	ix
List of Tables	xv
List of Figures	xx
Abbreviations	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	2
1.3 Outline	3
2 Background	5
2.1 Additive manufacturing (AM)	5
2.1.1 Manufacturing processes	5
2.1.2 Applications	6
2.1.3 Technologies	7
2.1.4 Gantry systems	9
2.1.5 Processing pipeline	11
2.2 Robot technology	14
2.2.1 Terminology	15

2.2.2	Industrial robots in AM	17
3	Literature study	19
3.1	Challenges in AM	19
3.1.1	Large scale	20
3.1.2	Support structures	21
3.1.3	Quality of part	22
3.2	AM by robot manipulator	24
3.2.1	Multi-robot systems	24
3.2.2	Large-scale structures	24
3.2.3	Support reduction methods	25
4	Robotic AM system	27
4.1	Original problem formulation	27
4.2	Processing pipeline	29
4.2.1	Obtain digital model	31
4.2.2	Sampling	31
4.2.3	Accumulate voxels into printing layers	31
4.2.4	Calculate tool-paths on each layer	32
4.2.5	Translate tool-paths to machine-readable instructions	32
4.3	Overview of layer generation method	33
4.3.1	Greedy scheme	34
4.3.2	Improvement scheme	34
4.4	Convex front advancing (CFA)	34
4.4.1	Understanding the results	35
4.4.2	Notation	36
4.4.3	Greedy scheme	36
4.4.4	Example: GCFA	39
4.4.5	Improvement scheme	48
4.4.6	Example: GCFA-ISP	50
4.5	Hardware	57
5	Implementation and results	59
5.1	Implementation	59
5.1.1	Dependencies	59
5.1.2	Pre-processing of object	61
5.1.3	Convex front advancing	64
5.2	Results	68
5.2.1	Test objects	68
5.2.2	Presenting the results	69

5.2.3	Summary of results	98
6	Discussion	99
6.1	Changes made to the original algorithms	99
6.1.1	Defining AM-stable neighbors (ASN)	99
6.1.2	ISP voxel accumulation threshold	100
6.2	Simulation results	101
6.2.1	Note: 2D vs. 3D objects	102
6.2.2	Looking at the result tables	102
6.2.3	Interpreting the results	104
6.2.4	Evaluation of the results	105
6.2.5	How to improve the results	106
6.3	Curved layers vs. planar layers	106
6.4	Hardware	110
6.4.1	Fixed nozzle	110
6.4.2	Fixed platform	111
6.4.3	Fixed, elevated platform	112
6.4.4	Summary	112
7	Conclusion and future work	115
7.1	Conclusion	115
7.2	Future work	116
7.2.1	Physical experiments	118
A	Mathematical formulas	119
B	Algorithms	121
C	Code	125
	Bibliography	127

List of Tables

5.1	Algorithm performance in terms of missed voxels.	98
5.2	Algorithm performance in terms of manufacturing layers.	98
6.1	An overview of where to find which simulation result in the thesis.	102
6.2	Comparison of algorithm performance in terms of missed voxels . .	103
6.3	Comparison of algorithm performance in terms of manufacturing layers	104
6.4	Evaluation of test results	105
6.5	Identification of the worst results obtained using one of the new methods for each object.	107
6.6	Comparing number of missed voxels of the worst result obtained using the new method and uniform slicing.	109
6.7	Overview of which hardware obtains what properties.	113

List of Figures

2.1	An object being fabricated by an AM technology.	6
2.2	Cartesian coordinate system	9
2.3	Schematic drawing of a common 3DOF 3D printer using the gantry configuration	10
2.4	A digital model	11
2.5	Changing build direction can influence the need for support structures. Assuming fixed, vertical build direction, this object can be rotated to eliminate the need for added support	13
2.6	Uniform division of object.	13
2.7	The process of obtaining a 3D object, dividing it into layers and covering the layers with material-paths.	15
2.8	Schematic drawing of a 6DOF robot manipulator. $z_1 - z_6$ denotes the axes of rotation of the robot.	16
2.9	Improving the gantry system by giving the deposition tool the ability to change its orientation and thereby the ability to change the build direction for manufacturing.	17
2.10	A robotic AM system can be made by mounting a deposition tool onto any robot.	18
3.1	An example situation of an object containing a large overhang solved by adding support structures.	21
3.2	The staircase effect	23
3.3	Improving surface quality by covering an object with a single curved layer of material.	24
4.1	Methods for dividing a general object into a sequence of manufacturing layers proposed in Dai et al. (2018).	29
4.2	Pipeline of the AM process proposed by Dai et al. (2018).	30
4.3	Sampling process of a 3D object.	32

4.4	Implementation scope of the thesis.	33
4.5	The part of the pipeline of the robotic AM system we will study in this section.	35
4.6	Color mapping of layers in growing field.	35
4.7	AM stable neighbors (ASNs) of the center point, as defined in Definition 3. As long as the center point have been previously manufactured using a rapidly curing material, each ASN(center point) is guaranteed to be manufactured in a support-free manner.	38
4.8	The convex hull is a convex polyhedron and can be viewed as a conservative accessible surface.	40
4.9	This is the example object that will be used in the curved layer generation examples.	41
4.10	The first step of the algorithm is to generate the initial current manufacturing layer $\mathcal{L}_c = \mathcal{L}_1$. All voxels adjacent to the platform is added to \mathcal{L}_1	42
4.11	The process of finding the set of ASNs of a voxel.	43
4.12	Here we see the second layer output by the greedy CFA.	43
4.13	Layer 3 are generated in the same manner as layer 2.	44
4.14	Layer 4 are generated in the same manner as layer 3.	44
4.15	Accumulation of voxels in critical region.	45
4.16	The first occurrence of a curved layer, \mathcal{L}_{10}	46
4.17	The first occurrence of shadowed voxels. The red crosses show voxels that cannot be added to the next layer due to being inside the current convex front. The algorithm have no means to prevent voxels from being shadowed.	46
4.18	The resulting growing field \mathcal{G} of the GCFA example	47
4.19	We use the same example object for the GCFA-ISP example as in the GCFA example in Section 4.4.4. As long as no voxels are shadowed by the sequence of convex fronts, the ISP improvement algorithm do not interfere with the greedy output.	50
4.20	If \mathcal{L}_{10} is chosen as it is here, two voxels will be shadowed when determining the next layer by the greedy approach. An attempt should be made to "save" these voxels before continuing the program.	51
4.21	Improving the layer proposed by the greedy scheme using ISP.	54
4.22	comparing the generation of the 10th layer by the greedy approach and by the greedy approach with improvement.	55
4.23	The resulting growing field \mathcal{G} of the GCFA-ISP example.	56
4.24	Comparing the resulting growing fields.	57
4.25	Schematic drawings of the hardware from Dai et al. 2018	58

5.1	Two different ways of representing a 2D object.	62
5.2	Two different ways of representing a 3D object. For simplicity, the object is decomposed into $n = 4$ two dimensional sub-objects. . . .	63
5.3	Interface of the implemented CFA program.	64
5.4	Information flow of the GCFA-implementation	66
5.5	Information flow of the GCFA-ISP-implementation	67
5.6	Objects used for testing the algorithms from Chapter 4	68
5.7	Test object 1 represented in three ways.	69
5.8	Object 1 - growing field \mathcal{G} with respect to small working platform.	70
5.9	Object 1 - growing field \mathcal{G} with respect to medium working platform.	71
5.10	Object 1 - growing field \mathcal{G} with respect to large working platform.	72
5.11	Test object 2 represented in two ways.	73
5.12	Object 2 - growing field \mathcal{G} with respect to small working platform.	74
5.13	Object 2 - growing field \mathcal{G} with respect to medium working platform.	74
5.14	Object 2 - growing field \mathcal{G} with respect to large working platform.	75
5.15	Test object 3 represented in two ways.	76
5.16	Object 3 - growing field \mathcal{G} with respect to small working platform.	77
5.17	Object 3 - growing field \mathcal{G} with respect to medium working platform.	77
5.18	Object 3 - growing field \mathcal{G} with respect to large working platform.	78
5.19	Test object 4 represented in two ways.	79
5.20	Object 4 - growing field \mathcal{G} with respect to small working platform.	80
5.21	Object 4 - growing field \mathcal{G} with respect to medium working platform.	80
5.22	Object 4 - growing field \mathcal{G} with respect to large working platform.	81
5.23	Test object 5 represented in two ways.	82
5.24	Object 5 - growing field \mathcal{G} with respect to small working platform.	83
5.25	Object 5 - growing field \mathcal{G} with respect to medium working platform.	83
5.26	Object 5 - growing field \mathcal{G} with respect to large working platform.	84
5.27	Test object 6 represented in two ways.	85
5.28	The result from running the GCFA program on object 6 together with a medium working platform.	86
5.29	Axes of symmetry of growing fields. Object seen from above	86
5.30	Results of GCFA w.r.t. working platform of medium size. Each sub-figure show a cross section of the growing field at constant y -values.	88
5.31	Results of GCFA w.r.t. working platform of medium size. Each sub-figure show a cross section of the growing field at constant z -values.	89
5.32	The result from running the GCFA program on object 6 together with a large working platform.	90

5.33	Axes of symmetry of growing fields. Object seen from above	90
5.34	Results of GCFA w.r.t. working platform of large size. Each sub-figure show a cross section of the growing field at constant y -values.	93
5.35	Results of GCFA w.r.t. working platform of large size. Each sub-figure show a cross section of the growing field at constant z -values.	94
5.36	The result from running the GCFA-ISP program on object 6 together with a large working platform.	95
5.37	Results of GCFA-ISP w.r.t working platform of large size.	96
5.38	Results of GCFA w.r.t. working platform of large size. Each sub-figure show a cross section of the growing field at constant z -values.	97
6.1	Results using two different choices of norm-values when defining the set of ASNs.	100
6.2	Object 1 – The worst result with new method vs. uniform slicing .	108
6.3	Object 2 – The worst result with new method vs. uniform slicing .	108
6.4	Object 3 – The worst result with new method vs. uniform slicing .	108
6.5	Object 4 – The worst result with new method vs. uniform slicing .	109
6.6	Object 5 – The worst result with new method vs. uniform slicing .	109
6.7	Hardware where the platform is attached to the wrist of the robot arm and the manufacturing tool is attached to an overhead beam.	110
6.8	Hardware where the platform is on the floor and the manufacturing tool is attached to the wrist of the robot arm.	111
6.9	Hardware where the platform is attached to an elevation-device and the manufacturing tool is attached to the wrist of the robot arm. .	112

Abbreviations

2D	Two dimensional
3D	Three Dimensional
AM	Additive Manufacturing
ASN	AM-stable neighbor
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CFA	Convex Front Advancing
CNC	Computer Numerical Control
FDM	Fused Deposition Modeling
FFF	Fused Filament Fabrication
FM	Formative Manufacturing
GCFA	Greedy Convex Front Advancing
GCFA-ISP	Greedy Convex Front Advancing with Incremental Shadow Prevention
ISP	Incremental Shadow Prevention
PFG	Peeling Field Generation
SM	Subtractive Manufacturing



Chapter 1

Introduction

1.1 Motivation

This thesis is written for a research project that is looking for ways to make sustainable manufacturing possible in high-cost countries. The research project is part of SFI Manufacturing, a cross-disciplinary centre for research based on innovation for competitive high-value manufacturing in Norway [1].

In order to keep consumer prices low, parts are often produced in high-volume by large factories, often in low-cost countries. Global wealth is increasing [2], which may lead to an even higher demand for parts and products. A byproduct of manufacturing is often large amounts of production-waste. Waste, both in terms of reduction and recycling, is an environmental concern because the waste that ends up astray can cause serious environmental damage and the recycling of waste requires energy. By making the production of specialized parts in low volume more affordable, mass production could lose some of its popularity, which could lead to a reduction of the amount of waste produced.

Because low-cost countries can offer production at a very low unit price when the batch size is high enough, the relative difference between the item-production cost and the consumer-price of the item becomes large. This happens because the item is produced in a different market than it is sold, which often makes the choice of moving production abroad a profitable one. Worldwide shipping of products cause pollution of the environment. Developing methods that enable local production is one way of reducing the amount of pollution.

Additive manufacturing (AM), more commonly known as 3D printing, is a

new manufacturing technology that enables for geographic independent manufacturing, i.e. the realization process of a part can take place anywhere, all that is needed is a digital model of the part. It is also a manufacturing technology well suited for low volume production of specialized parts. This means that developing methods for AM that enable the technology to manufacture general parts of all shapes and sizes, i.e. development for it to becoming a *general* manufacturing technology, can contribute to a solution to the above-mentioned problems.

In this thesis, AM is investigated in application together with robot technology. Taking advantage of the large work-space and the ability for an industrial robot to perform high-precision, repetitive tasks, robotic AM can be a mean for the manufacturing technology to be more applicable in many applications than the technology currently is today.

1.2 Contributions

In this section, all contributions made by the author during her work with this thesis is presented. The contributions are:

- Implementation of two methods for decomposing 2D and 3D objects into curved manufacturing layers for AM based on algorithms from Dai et al. (2018), referred to as GCFA method and GCFA-ISP method, have been performed.
- The algorithms were reviewed and altered in order to generate satisfying and realizable results from simulation.
- Suitable test objects were generated and the implemented algorithms were tested on these objects.
- From repeated testing of the algorithms, properties that impact the decomposition of the object and the quality of the result were identified.
- In addition to evaluating the results based on voxels count, evaluation is also performed based on the number of manufacturing layers generated.
- The methods for generating curved layers have been compared to not only each other, but also to an existing object decomposition method commonly used in AM.
- Different types of robotic AM hardware have been evaluated and compared, and based on this, a new hardware, which is a combination of the hardware used in Dai et al. (2018) and an other common robotic AM hardware, was proposed.

1.3 Outline

Chapter 2 gives some background information on important topics of the thesis. This includes some basic theory on AM and robot technology, and how industrial robots can be used to extend the area of application of AM. Chapter 3 presents findings from literature reviews identifying challenges in AM and solutions to these challenges. Chapter 4 investigates a robotic AM system newly proposed in research [3] with a focus on a decomposition method for dividing objects into curved manufacturing layers that cannot be realized using traditional AM configurations. A decomposition method is implemented and Chapter 5 give the implementation details before presenting some simulation results from testing the implemented program. Chapter 6 discusses the findings of the thesis, including the algorithms from the paper, simulation results, and hardware for realizing objects. Chapter 7 makes some concluding notes and suggestions for future work.

Chapter 2

Background

The goal of this chapter is to give the reader some knowledge on the topics this thesis will touch upon. First, additive manufacturing (AM), including applications, different technologies, and hardware, is presented in Section 2.1. Then, in Section 2.2, some important robot terminology is presented, followed by how robots can be applied to AM to improve the abilities of the manufacturing technology.

The content of this section is taken from a preliminary project thesis written by the author of this thesis [4]. Some illustrations have been added to demonstrate important concepts.

2.1 Additive manufacturing (AM)

AM is defined as the "process of joining materials to make parts from 3D model data, usually layer upon layer" [5] and is a relatively new manufacturing technology. In 1981, the Japanese researcher Hideo Kodama performed and documented experiments showing that solid models of complex shapes could be fabricated by stacking cross-sectional layers [6]. This was the first documented attempt on using an AM technology.

2.1.1 Manufacturing processes

In literature, AM is often referred to as 3D printing. Other names used is layered manufacturing [7], additive layer manufacturing [8], solid free-form fabrication [9] [10] and rapid prototyping [11]. In Figure 2.1, the manufacturing principle of AM is illustrated. AM is characterized by the stacking of layers of material. In the

figure, the current layer is being fabricated by the rightward movement of the manufacturing tool and the previously manufactured layers can be seen below the current layer.

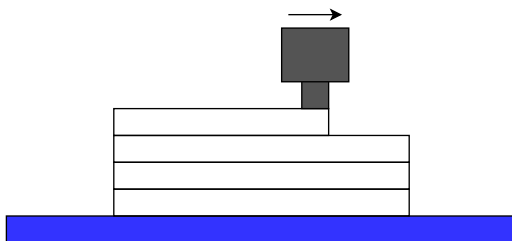


Figure 2.1: An object being fabricated by an AM technology.

AM makes up the third out of three common manufacturing technologies of today. The formal terms of the other two technologies are subtractive manufacturing (SM) and formative manufacturing (FM).

SM works in the opposite way of AM. In AM, a part is made by adding material, while in SM, material is being subtracted from a piece of material larger than the part itself. The latter method result in large amounts of production waste, since material is constantly being removed. If the part is made from a material that cannot be, e.g., recycled into new products, this is an important concern.

In FM, a part is realized by filling a mold with material and let the material cure inside the mold. A disadvantage to this method is that it needs a specific mold for each specific part. In addition, each mold can only be used to make a finite amount of parts before it is worn down and must be replaced.

Neither SM nor FM is well suited for the production of parts consisting of more than one material. They are neither very well suited for the production of low quantity, highly specialized items.

2.1.2 Applications

AM offers the potential for developing complex, customized products that are expensive to produce with other techniques [12]. If the market environment is characterized by uncertainty, high product variety or fluctuating customer tastes, the firms that are equipped with flexible manufacturing technologies such as AM

may obtain an important competitive advantage [13]. Four patterns that characterize markets for AM have been identified in research:

1. Small production output
2. High product complexity
3. High demand for product customization tailored to individual customer needs
4. Spatially remote demand for products

The technology was developed to aid engineers and designers in making new products, primarily through prototyping [14]. Through recent advances within the technology, it is also developing into being well suited for manufacturing of end-use parts. Companies such as materialise¹ delivers 3D printing services on areas such as aerospace, aeronautics, automotive and healthcare. Automotive, medical, aerospace and military industries are all industries that require high precision and reliability, and they are all industries that benefit from AM technologies [15].

2.1.3 Technologies

AM started to emerge in research in the 1980s and the technology was developed by many research teams in parallel. As a result of the various research engagements, we today have many different types of AM technologies, all sharing the same principle of stacking layers of material. We will now have a look at different AM technologies, some specifications and how to best separate the technologies from each other.

Classification

Due to the extensive research work made on the field of AM, many technologies exist today. Therefore, many contributions have been made with goal of categorizing and classifying the various technologies. The technologies may be classified according to:

- Material property [16]:
 - Liquid-based
 - Powder-based
 - Solid-based
- Functional similarities [17]

¹www.materialise.com

- Material curing [18]:

Deposition of material
Solidification of material

The way the technologies will be classified in this thesis is by dividing them into deposition- and solidification methods. In material solidification methods parts are realized by the solidification of a non-solid material, such as powder or liquid, typically within a tank. This class of AM technologies will not be in the scope of this thesis.

Material deposition methods works by locally deposit material onto a plane or onto already cured material to create a new layer. We further divide the deposition technologies into three new categories:

- Material extrusion
- Material jetting
- Directed energy deposition

Scope

The scope of this thesis will be on extrusion-based deposition technologies. The reason why this is chosen as the scope is because the thesis will study the application of extrusion AM together with industrial robots, where a deposition tool will be mounted on the arm of the robot.

Material extrusion

Material extrusion systems are based on the principle that material contained in a reservoir is forced out through a nozzle when pressure is applied [14]. A majority of the AM processes developed for polymers and polymer composites are extrusion-based processes [19]. If both nozzle speed and pressure are constant, and the material is in a semi-solid state, the system will make a cylindrical path of material with constant diameter. The most common extrusion-based techniques are Fused Deposition Modeling (FDM) [20] patented by company Stratasys², or Fused Filament Fabrication (FFF) [21] which is a similar method that is not patented by any company [14]. The experiments in this thesis will be based on, but not limited to, an FDM/FFF system.

²www.stratasys.com

In the material deposition technologies, the material can only be deposited on a previously deposited layer, meaning that there is a strong need for added support structures, which is a drawback of this method [18]. The material deposition techniques have the advantages of the ability to combine multiple materials. The printing time in deposition methods is mostly dependent on the part volume.

2.1.4 Gantry systems

Most AM facilities have a closed fabrication platform which limits the size of the object to be printed [22]. The most common configuration of an AM machine is the gantry configuration. In a gantry system, the manufacturing tool is mounted onto an overhead system, often a narrow beam. The manufacturing tool is attached to the beam and together with the beam, it moves to a sequence of specified points in space, thus producing the part point by point.

Cartesian coordinate system

The number of axes a system can freely move along or rotate around is often referred to as the dimensions of the system or as the systems degree of freedom (DOF). The gantry system uses a Cartesian coordinate system of three dimensions. A Cartesian coordinate system specifies points uniquely in space by a set of numerical coordinates. The coordinates are signed distances to the point from three fixed orthogonal axes [23]. An illustration of a three-dimensional Cartesian coordinate system can be seen in Figure 2.2.

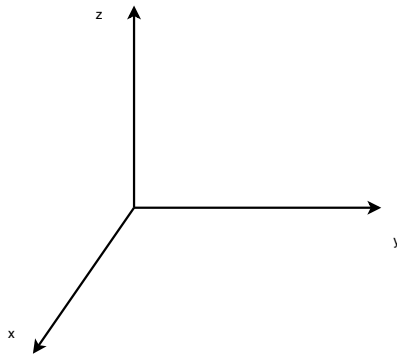


Figure 2.2: Cartesian coordinate system

Figure 2.3 shows a gantry system resting at two different positions. The arrows illustrate the freedom of movement. We see that the system can move freely in space along the x -, y - and z -directions, so this can be referred to as a 3 DOF system. A system of this configuration has no way to change the orientation of the tool, only the position. A consequence of this is that the printing direction, here z , is fixed.

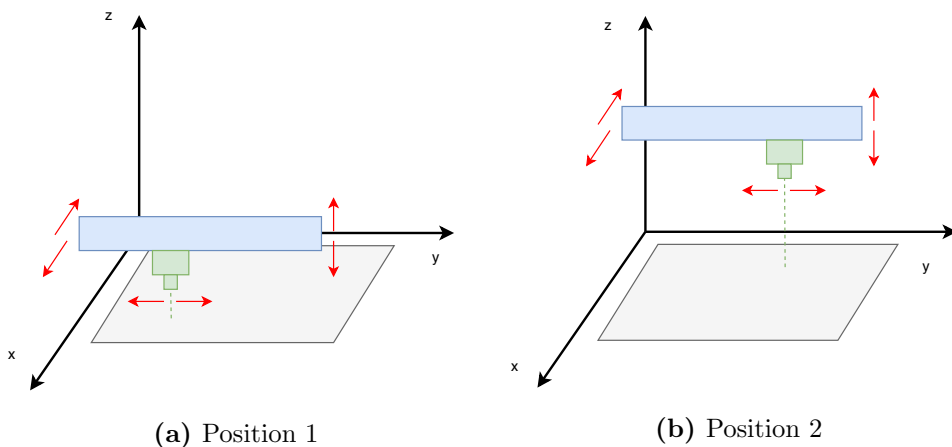


Figure 2.3: Schematic drawing of a common 3DOF 3D printer using the gantry configuration

The chamber- or tank volume is determined by the limits of each direction of the different movements, e.g., $\min(x)$ and $\max(x)$ are the limits of the movement in x -direction, and thus contributes to determining the chamber volume, i.e. the workspace of the machine.

Limitations

Three challenges that arise from using 3 DOF gantry systems are:

- the system is typically high-weight [24]
- the size of the part to be manufactured is constrained by the volume of the internal workspace [24]
- the printing direction is fixed so that support structures must be added [3]

The above challenges will be discussed in greater detail in Section 3.1. Common solutions for these problems are often based on either scaling up the size of

the whole system so that the chamber- or tank volume gets larger or to decompose the object before printing and have a separate assembly phase after manufacturing. However, these are not very flexible solutions for the additive fabrication of large scale structures.

2.1.5 Processing pipeline

Independent of which AM technology used, a product must often go through many of the same steps to become fully realized and operational. This section reviews the AM processing pipeline, i.e. the steps an object usually must go through during fabrication by an AM technology. Different technologies may require more or less attention for a number of the steps [14].

Step 0: Obtain a digital model

Computer Aided Design (CAD) is defined as the use of computer systems to assist in the creation, modification, analysis or optimization of a design [25]. The input to a digital manufacturing process such as AM is a digital model of an object. We will refer to this digital input model as a CAD model. In this section we will consider the digital model in Figure 2.4. The model is often generated by a CAD software. An alternative is to use reverse engineering to realize a physical model in CAD, e.g., laser scanning of a real object [14] or using data obtained by a medical imaging system.

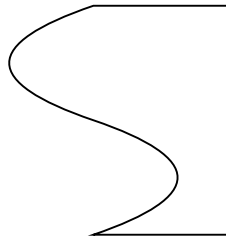


Figure 2.4: A digital model

Step 1: CAD to Computer-Aided Manufacturing (CAM) representation

The input to the manufacturing process often needs to be a geometry description of the data in the CAD model, so the CAD model needs to be converted into the

correct format. This conversion outputs what is often referred to as a Computer-Aided Manufacturing (CAM) model [18].

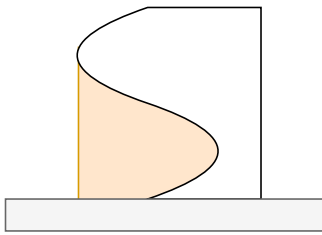
The conversion from CAD to CAM can be done using a variety of methods. One common method is to use tessellation. Tessellation is a method to approximate the surface of a 3D body [26] using a mesh of triangles [27]. The tessellated model takes the file format .stl and is accepted by most 3D printers [14]. It is important to note that tessellation outputs an approximation of the CAD model. The accuracy of the approximation of the model can be adjusted by the size of the tessellation triangles, smaller triangles will give a more accurate representation. An alternative method that gives a mathematically accurate description of the geometry data is the STEP file format [27].

The resolution of the realized model is dependent on both the resolution of the approximated CAM model and the dimension of the deposited material, which again depends on the size and velocity of the deposition tool.

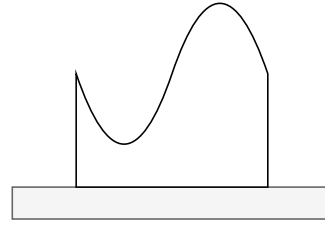
Step 2: Determine build direction and generate support structures

In most AM technologies, the build direction is fixed. So after the input model has been converted from CAD to CAM, the build direction must be determined. Since AM is performed layer by layer, the build direction is crucial and directly influences important factors such as build time, the necessity of support structures, surface quality and the functionality of the part [18]. Figure 2.5 demonstrates how much the orientation of the object can influence the need for support structures. Figure 2.5a show the object from Figure 2.4 with its original orientation. Assuming horizontal build direction, the marked area in the figure illustrates the required support structures in this case. If the object was rotated 90° clockwise, see Figure 2.5b, the need for support structures would be completely eliminated.

When the optimal build direction has been determined, support structures are added to stabilize critical regions.



(a) With this orientation, support structures are needed in order to successfully manufacture the object.

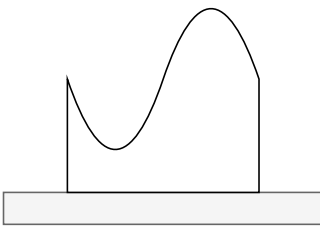


(b) Changing orientation can reduce or eliminate the need for support structures.

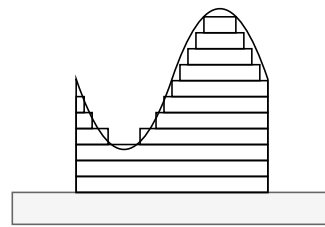
Figure 2.5: Changing build direction can influence the need for support structures. Assuming fixed, vertical build direction, this object can be rotated to eliminate the need for added support

Step 4: Divide the object into manufacturing layers

This is the step where the CAM model is divided into a set of individual layers, sometimes called slices [18], which are to be covered with material. The layers are usually restricted, but not limited, to be planar. Assuming for now that we are slicing the model into planar layers, most AM software applies algorithms that output a uniform division of the object, meaning that all layers have the same thickness or height. An illustration of uniform slicing can be found in Figure 2.6. The thickness of the layers influences the appearance of the realized object.



(a) After the object orientation is determined, the object is ready to be divided into manufacturing layers.



(b) The object has been divided into layers of uniform height.

Figure 2.6: Uniform division of object.

Step 5: Determine tool paths to cover the layers

When the object has been divided into a sequence of layers for manufacturing, the path for the deposition tool to move along can be calculated. The tool-paths determine how the material is distributed on each layer. To realize the object, each layer must be filled with material and the job for a path generating algorithm is to find a path that connects all the points in each layer with no overlapping of the paths. The path need not be continuous, but it should not intersect with itself at any point.

Step 6: Machine instructions and post processing

The final step of the process planning is to determine the machine instructions that the fabrication tool must execute to build the part [18]. Post-processing may include removal of support structures, machining or assembly of individual parts to obtain the finished product.

Summary with example

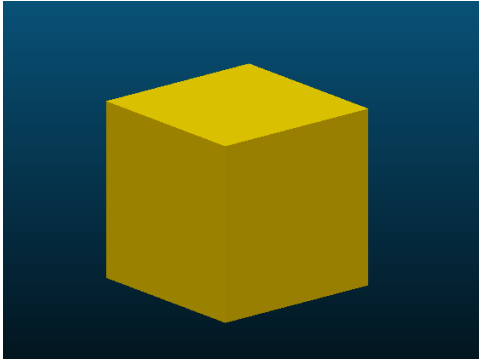
Figure 2.7 shows the screenshots of the AM pipeline steps and serves as a summary of the steps reviewed in this section.

First, we need an object to realize. Using the free CAD software FreeCAD, the object in Figure 2.7a is created. The screenshot is from the view-mode in the AM software slic3r, which the CAD model has been exported to after created. Using the option of uniform slicing, the slic3r outputs the divided object in Figure 2.7b. The height of the layers can be chosen freely and affects, among other things, the appearance of the object. Since this object is a cube with only horizontal or vertical surfaces, the layer height or thickness should be chosen according to which hardware is available.

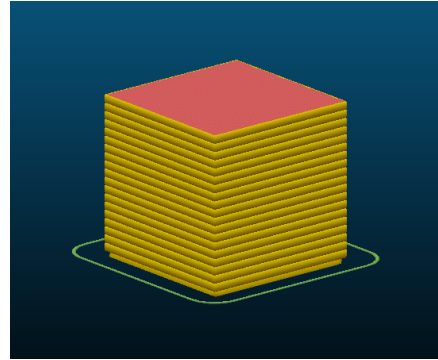
The software lets us choose among many material patterns to cover the surface. The pattern chosen in this example can be seen in Figure 2.7c and is called "concentric" tool-paths. In Figure 2.7d we see how the tool paths will look on the object.

2.2 Robot technology

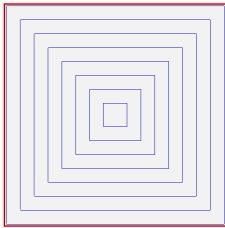
In this thesis, we wish to investigate the benefits of using robot technology in an AM context. Robot technology is a broad term, so to determine which sub-group



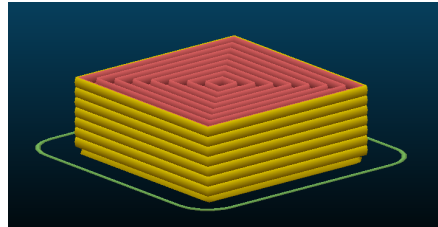
(a) Input to the manufacturing process created with FreeCAD.



(b) After build direction is determined, the model can be divided into layers. Slicing is performed in slic3r.



(c) Layers covered with concentric tool-paths.



(d) Layers covered with tool-paths is stacked on top of each other to create the object.

Figure 2.7: The process of obtaining a 3D object, dividing it into layers and covering the layers with material-paths.

of the technology we will consider moving forward, we need to first establish some robot terminology. This is followed by a discussion on the benefits of using robots in AM applications.

2.2.1 Terminology

Robot Institute of America (RIA) defines a robot as a "reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks". In this thesis the term robot will refer to a mechanical arm operating under computer control, sometimes called a computer-controlled industrial manipulator [28]. Figure 2.8 show a schematic drawing of one such robot. The book

”Robot modeling and control” by M. W. Spong and colleagues [28] defines some robot terminology. Some important concepts is presented in short below.

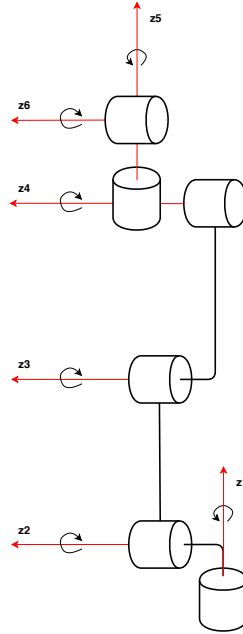


Figure 2.8: Schematic drawing of a 6DOF robot manipulator. $z_1 - z_6$ denotes the axes of rotation of the robot.

An industrial robotic manipulator is made up of *joints*, either revolute or prismatic with *links* connecting them. To be of any practical use, the robot also needs an *end effector* to interact with the environment. An end effector is the device or tool attached to the end of the robot arm [29]. It can take any shape or form, and in AM it is often a printing nozzle.

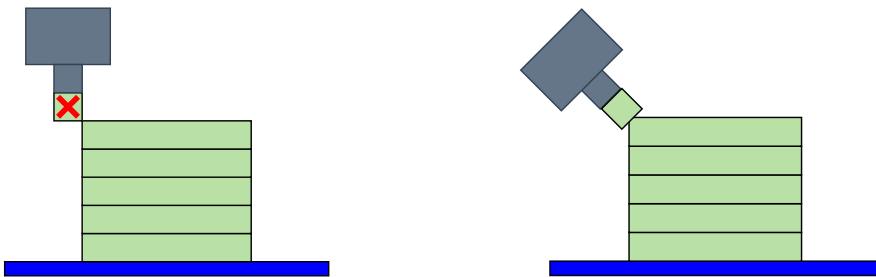
We are often interested in describing the *position* of the end effector together with its *orientation* to specify the full configuration of the tool. The links and joints of a robot manipulator form a kinematic chain and the position is described in terms of the joint variables of the joints that make up the arm. In the robot arm in Figure 2.8 the variables (z_1, z_2, z_3) specify the position of the tool while the variables (z_4, z_5, z_6) specify the orientation. The *workspace* of a manipulator is the total volume made up of all single point the end effector can reach.

A system is said to have n *degrees of freedom* (DOF) if its configuration can be specified with a minimum of n parameters. For a robot manipulator, we may count the number of joints variables to deduce the number of DOFs. Since the robot arm in Figure 2.8 have 6 joint variables, $z_1 - z_6$, the robot arm is said to be a 6 DOF robot. To be able to reach any rigid object in the workspace, the robot needs to possess at least 6 DOFs.

2.2.2 Industrial robots in AM

The 3 DOF gantry system from Section 2.1.4 have limitations that prevent AM from becoming a general manufacturing technique. Some refer to using this gantry system as 2.5D in stead of 3D printing, referring to the manufacturing being limited by the fixed build direction [3].

The build direction is determined by the orientation of the tool. Figure 2.9 show the manufacturing of an overhang. In Figure 2.9a, a fixed vertical build direction cause the deposited material to not stick to the previous layer when trying to manufacture a layer that surpass the previous layer in length. If the tool have the ability to change its orientation, we can get the improved situation in Figure 2.9b. Here we see that by rotating the deposition tool by 45° , material can be applied to the previous layer and also stick to it. An assumption made in this scenario is that the deposited material must be rapidly curing and in a semi-fluid state.



(a) Tool angle = 0° . It is impossible to manufacture this point in a support-free manner using this build direction.

(b) Tool angle = 45° . Assuming a good material, the material will not need additional support in this case.

Figure 2.9: Improving the gantry system by giving the deposition tool the ability to change its orientation and thereby the ability to change the build direction for manufacturing.

The scenario in Figure 2.9 show that changing the orientation of the tool can increase the region of self-supporting material accumulation, thus making the technology more flexible by utilizing more degrees of freedom. If the tool is able to change its orientation, material can not only be accumulated strictly on top of previous material, but also in the neighborhood of the material. This happens because a printing tool at an angle could give just enough support for the new material to stick to the already cured material.

We will now have a look at how the concept of expanding the region of self-supportive material accumulation, demonstrated above, can be utilized to make an improved AM system using a 6 DOF robot manipulator such as the one in Figure 2.8.

By mounting a manufacturing tool onto the wrist of the robot arm, we obtain an AM system that have the ability to freely change the build direction at any point in time during fabrication. The flexibility in the movement of the arm makes the workspace larger than the arm itself which means that the system can access regions that are not possible for a gantry-based machine [30]. Figure 2.10 show a complete 6 DOF robot based AM system. We see the deposition tool attached to the end of the arm and and the manufacturing platform below.

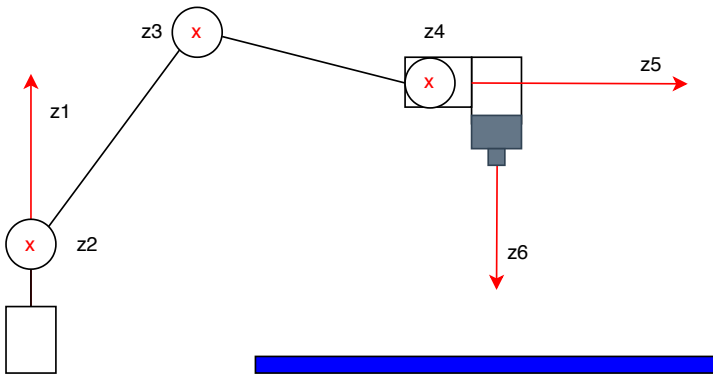


Figure 2.10: A robotic AM system can be made by mounting a deposition tool onto any robot.

Chapter 3

Literature study

In the following chapter, findings from literature reviews on important topics are presented. Section 3.1 present results obtained from a literature review performed by the author with aim of identifying solutions to challenges in AM that prevents the technology from becoming a general manufacturing technology. Section 3.2 present results of a literature review on the use of robot technology in AM and what challenges it has been used to solve and how.

The literature reviews were performed during a preliminary project preceding to this thesis [4] so the content of this chapter is not an original work of this thesis, but is presented here in order to give the reader an overview of the challenges of using AM as a general manufacturing technology and how the technology can be improved, and has been improved, by the use of robot manipulators. Changes have been made to the original material and less important subjects have been removed so that it fits the purpose of this work.

3.1 Challenges in AM

We have already identified some limitations from using AM to realize products. Many of the problems arise when using the 3 DOF gantry system from Section 2.1.4, which is the most common AM configuration for all AM technologies, some of them presented in Section 2.1.3. We start by repeating the identified limitations from Section 2.1.4:

- Often high-weight system
- Size of manufactured product
- Fixed build direction

The research team in [12] present a timeline of significant developments for the use of AM techniques within different groups of society. As the manufacturing discipline has gained more supporters, the amount of research on topics related to AM and the challenges of AM in different contexts have increased considerably.

In this section, challenges related to fabricating a part using an AM technology will be reviewed together with some proposed solutions from research. Literature reviews to uncover solutions to the below challenges have been performed:

- The size of the object
- The minimization of support structures
- The quality of the finished part

Even though research teams often aim to solve one isolated problem, sometimes the solution gives improved results in other areas as well. The true flexibility of AM cannot, in general, be fully benefited from until solutions to all, or some of the AM challenges have been found.

3.1.1 Large scale

AM technologies are normally used to make small components, on the "desktop" scale [31]. This is because, as explained in Section 2.1.4, the printing volume is often constrained by the configuration of the 3 DOF gantry system. The total weight of the structure is also a factor to consider here.

Large scale AM is a relatively new field of research, but the advantages of AM, discussed in Section 2.1.2, are still present at large scale [32], so the topic should be studied further. The main application of large-scale AM, and where most of the research is focused, is in the automation of construction processes industry [32]. In construction, every structure is often unique in dimension so traditionally either standard size materials are cut down to fit specifications or molds are created to form each component [31]. Companies are now exploring the market's interest in 3D printed buildings.

The research team in [32] propose a large-scale 3D foam printing system that uses a 6 DOF cable-suspended parallel mechanism for positioning which can construct any 3D geometry. A gantry-type system with added degrees of freedom is here used to control the position and the orientation of the deposition tool.

3.1.2 Support structures

Critical regions influence the fabrication of an object. Figure 3.1 shows an example situation. The object in Figure 3.1a contains an overhang forming a critical region. If this object is fabricated according to the uniform layers in Figure 3.1b, a situation will occur when manufacturing the layers from the fifth layer. Due to gravity, the layer material will fall to the ground instead of forming the layer. If not prevented or detected in real-time, the error will propagate to all successive layers. The critical region is highlighted in Figure 3.1c. The threshold for when the material will stick or not stick to the previous layer is material- and method-specific and is sometimes called a self-supporting angle [33]. To prevent the propagation of errors in the manufactured object, support structures are added beneath all overhanging areas of the critical region, see Figure 3.1d.

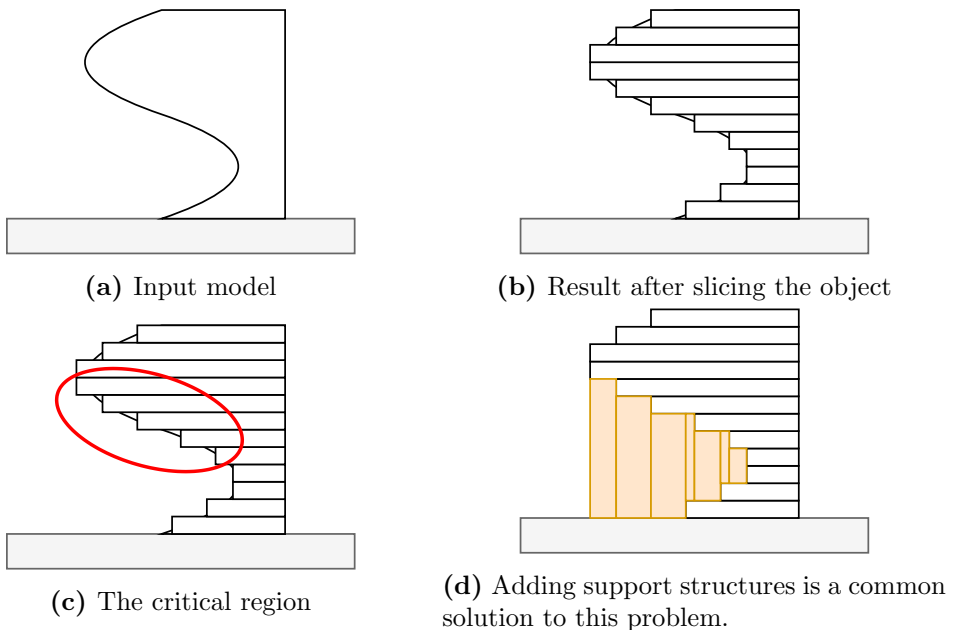


Figure 3.1: An example situation of an object containing a large overhang solved by adding support structures.

To limit the number of additional support structures needed for manufacturing, designers often manually change the shape of a design to make the individual parts of the object self-supportive. Hu et al. (2015) [33] tries to find a method to automate this design process and present an orientation-driven shape optimizer

to slim down the supporting structures used in single-material based AM. The method tries to deform an input model so that it has a shape that needs less support without losing important details of the original model. The method assumes that some deformation of the model is possible, which may exclude some important areas of applications and makes the method have a setback in terms of relevance.

Demir et al. (2018) [34] propose a divide-and-conquer approach for 3D printing which utilizes the properties of near-convexity. First, a model is decomposed into a low number of near-convex components. The components should all consist of only horizontal faces or faces with a larger angle than a printer-defined threshold. Letting the components be near-convex as opposed to convex reduces the complexity of the problem from being NP-hard and results in fewer components. The second phase is called the configuration phase, which aims to reduce printing time by laying the components out for printing so that all elements can be printed in one go. After manufacturing, the model is assembled from the individual components. The paper demonstrates that the approach reduces the consumption of printing resources and improves printing quality.

3.1.3 Quality of part

In extrusion-based techniques, the paths of the layered material can be considered as the building units of the process. Therefore it is natural that the properties and performance of the finished product are strongly affected by the tool-paths [35] which becomes an important factor in determining the quality of the part, both aesthetically and mechanically [8]. Many techniques have been developed for the production of high-quality 3D printed end-use parts, but more studies are still required to improve their systems and quality [15].

A general agreement in much research is that the mechanical properties of manufactured elements are closely related to which manufacturing technology was used and that it can vary significantly depending on production parameters such as printing temperature, velocity, and infill density [36]. For example, the adhesive strength between layers (or across filaments) of parts made by FDM is less than the strength of continuous filaments (longitudinal strength) [7], meaning that methods should aim to make parts from few layers instead of many small. Stava et al. [37] propose a method for detection and correction of major structural problems in 3D models before they are manufactured, while simultaneously minimize altering of the appearance.

Livesu et al. [18] defined the term fidelity as the degree of exactness with which a part has been reproduced starting from its design. Events like the staircase effect, see Figure 3.2, contribute to a reduced level of fidelity. The most common solution to avoiding this problem is to adjust the layer thickness. Other solutions do however exist in research. Thinner or adaptive height slices lead to a better finish, but also more time-consuming manufacturing, as production time is proportional to the number of layers.

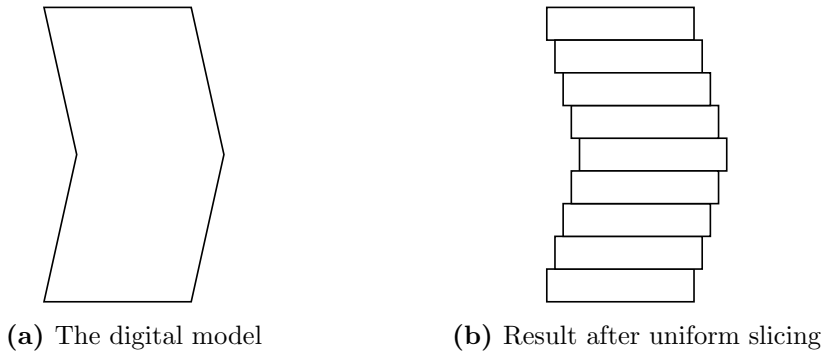
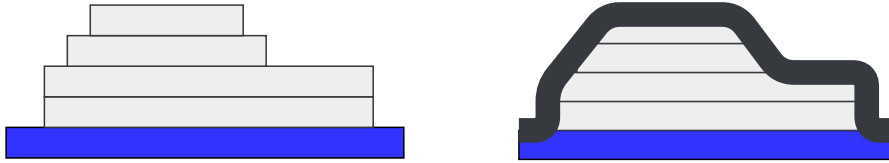


Figure 3.2: The staircase effect

Several papers have explored printing in curved layers as a means to improve the fidelity and the structural properties of a printed part. In Figure 3.3, a solution to the fidelity problem using curved layers is demonstrated. The solution is based on covering the rather rough-looking part with a single curved layer of material so that the sharp edges are all covered. Research teams have developed different methods for AM using curved layers [7] [8] [38].

Chakraborty et al. (2008) [7] presented a new technique developed for rapid prototyping named Curved Layer Fused Deposition Modeling (CLFDM) which they argue to be more suited than FDM in the manufacturing of thin, curved parts. By depositing material in curved non-horizontal layers using FDM, they investigate the manufacture of curved, thin parts. With small curvatures, this method may be realized using a 3DOF machine, but if the curvatures are large it would need 5DOF, such that the extruder axis can always coincide with the normal of the curved surface at the point to be manufactured. In Singamneni et al. (2010) [38], a project to build a machine capable of constructing a part by deposition of material as curved layers is presented. By modifying a Fab@Home desktop RP machine and developing algorithms, dynamic z-values were utilized..



(a) An reproduced object suffering from the staircase effect.

(b) The object no longer contain any sharp edges.

Figure 3.3: Improving surface quality by covering an object with a single curved layer of material.

3.2 AM by robot manipulator

With the help of additional degrees of freedom from e.g. a robot manipulator, new possibilities for 3D printing are being explored [39]. Robot arms allows for a part to be built along more than one direction [40], making the technology more flexible than it is traditionally. Section 2.2.2 explained how robot manipulators can be used to improve the results obtained by AM. In the following sections, some research on the use of robot manipulators in AM is presented.

3.2.1 Multi-robot systems

The use of robot manipulators in AM enables for the cooperation of more than one machine building the same part. Zhang et al. (2018) propose an AM system based on a team of mobile robots printing a single-piece concrete structure concurrently [41]. Another example of cooperating AM robots is the manufacturing of a metal bridge designed by Joris Laarman Lab [footnotewww.jorislarmann.com](http://www.jorislarmann.com) in cooperation with MX3D¹. The art- and construction project is installed over a canal in the city of Amsterdam, Netherlands.

3.2.2 Large-scale structures

We know that, unlike gantry systems for AM, the workspace of a robot manipulator can accommodate parts larger than itself [30]. This, in turn, enables for the fabrication of parts that are larger than the system hardware. The application

¹www.mx3d.com

of robot manipulators to enable fabrication of large scale parts have been studied and proof-of-concept results have been presented [42] [43].

A lot of the literature on the fabrication of large scale structures are concerned with applications in the field of architecture and construction. Due to the extensive cost and labor of building concrete walls, design for construction tend to be simple and repetitive in order to keep expenses at a low level [44]. Using AM to manufacture buildings and similar large scale structures would change to cost of manufacturing to be based merely on the cost of raw material rather than the geometric complexity [30].

Gosselin et al. (2016) present a FDM-like technique for layered manufacturing of ultra-high performance concrete [45]. The system consist of a deposition nozzle mounted on a 6 DOF robotic arm and demonstrates the ability for producing large-scale structures without temporary support. Keating and Oxman (2017) propose a system for constructing customized architectural-scale structures on-site [30]. The prototype of the system referred to as a Digital Construction Platform (DCP), is composed of a 4 DOF hydraulic arm and a 6 DOF robot arm, mounted on a mobile platform.

Hack et al. (2015) argues that cementitious materials are not ideal for fast, precise and geometrically unconstrained extrusion [46]. On the basis of this, they work on The Mesh Mould research project, which aim to develop a robotically fabricated construction system that allows for a cost and material efficient fabrication of geometrically complex concrete constructions. The construction system they propose is based on using AM fabricated large scale mesh structures as wall reinforcement.

3.2.3 Support reduction methods

Many papers on AM use decomposition methods to reduce the need for support structures. Taking advantage of the ability for robotic AM systems to change build direction, research teams propose different strategies to fabricate general CAD models, completely without or with reduced use of support structures [47] [39] [3]. Unlike the decomposition methods from Section 3.1.2 these methods do not require a separate assembly phase after manufacturing.

The basic idea of the work in two of the methods [47] [39] is to decompose a volume into segments and to find a "good" build direction for fabrication of each segment separately. Cutting planes are used to decompose the volume and

printing directions d_i perpendicular to each plane \mathcal{P}_i is found. Each segment is sliced in uniform layers and deposition of material can happen as in traditional methods. The robot arm is stationary during the fabrication of each segment. When a segment is finished, the robot changes configuration, and a new segment can be fabricated on top of the last segment using a new build direction.

The third method uses the same hardware as the two above, and is referred to as a two step method. First, a sequence of curved layers is extracted from the three dimensional volume. The layers is in the next step covered by tool-paths. The method uses curved tool-paths to produce each individual layer enabling a general volume with large overhangs to be printed in one session without added support structures. Here, the robot arm performs a continuous movement to manufacture all layers of the parts.

Chapter 4

Robotic AM system

In this chapter, a framework for robotic AM presented in the paper "Support-free volume printing by multi-axis motion" published in ACM Transactions on Graphics in 2018 [3] will be considered. The research team, Dai et al., propose a new robotic AM system for the support-free manufacturing of objects of arbitrary shape and present promising results from physical experiments. The method have previously been mentioned in Section 3.2.3 of this thesis.

The scope of this thesis will be on implementing and investigating two of the proposed algorithms for generating sequences of manufacturing layers. Section 4.1 gives the problem formulation of what is referred to as a dimension reduction problem. The processing pipeline, or framework, of the proposed solution to the formulated problem is presented in 4.2. Section 4.3 give an overview of one of the methods for solving part 1 of the dimension reduction problem is presented. This method decomposes an object into manufacturing layers. Section 4.4 review details of a layer generation method together with two working examples for demonstrating how the method works with and without improvement algorithm. The final section, Section 4.5, makes a presentation of the hardware used to produce the experimental results of the paper.

4.1 Original problem formulation

Dai et al. (2018) propose a framework for performing robotic AM using a 6 DOF robot manipulator with aim of automating the tool-path generation for multi-DOF AM on general models for support-free manufacturing. The approach was reviewed in Section 3.2.3 and is referred to as a dimension reduction strategy.

The dimension reduction strategy is based on dividing the problem into two sub-problems:

1. Decomposing the object into manufacturing layers
2. Covering the layers with material

The paper start by defining the decomposition- and covering problems:

Definition 1. (*Decomposition problem*) Given a solid model \mathcal{H} , we seek to decompose it into a sequence of (curved) surface layers $\{\mathcal{S}_{i=1,\dots,n}\}$ such as to represent the material accumulation in AM. This requires satisfying the following conditions:

1. The solid \mathcal{H} is well approximated by the curved layers as $\mathcal{H} \approx \cup_{i=1,\dots,n} \prod(\mathcal{S}_i)$ with $\prod(\mathcal{S}_i)$ denoting the convolution solid of \mathcal{S}_i by a sphere with diameter r (layer thickness), and there is no overlap between layers – $\prod(\mathcal{S}_i) \cap \prod(\mathcal{S}_j) = \emptyset (\forall j \neq i)$.
2. All surface patches \mathcal{S}_i are accessible – i.e., can be touched by a printer-head while not colliding with any $\prod(\mathcal{S}_k) (\forall k < i)$.
3. Every curved layers \mathcal{S}_i is enclosed by the dilation of previous curved layers, $\cup_{k=1,\dots,i-1} \prod(\mathcal{S}_k)$, with radius r – i.e., the overhang of \mathcal{S}_i is small so that an object under printing is self-supported.

Definition 2. (*Surface covering problem*) Given a curved layer surface \mathcal{S} that is feasible, we next consider how to efficiently generate a set of (curved) tool-paths $\{\mathcal{P}_j\}_{1,\dots,m}$ such that

1. We cover the layer: $\prod(\mathcal{S}) \approx \cup_{j=1,\dots,m} \prod(\mathcal{P}_j)$ with $\prod(\mathcal{P}_j)$ denoting the convolution solid of \mathcal{P}_j by a sphere with radius r , and there is no overlap between paths – i.e., $\prod(\mathcal{P}_i) \cap \prod(\mathcal{P}_j) = \emptyset (\forall j \neq i)$.
2. The number of curves, m , and the distance between the ending points of a tool-path, \mathcal{P}_j , and the starting point of the next tool path, \mathcal{P}_{j+1} , are minimized. This reduces the artifacts caused by spurious filaments (so-called *stringing*).
3. The shape of each curve \mathcal{P}_j should be as smooth as possible and be easily realized on a robotic arm.

The research team have developed methods in order to solve both problems formulated in Definition 1 and Definition 2. For the decomposition problem in Definition 1 two methods have been developed. One method is proposed for the covering problem in Definition 2. The layer generation methods proposed in the paper are:

- Greedy growing convex front advancing
 - with Incremental shadow prevention
 - with Adaptive refinement shadow prevention
- Peeling field generation

The approaches are referred to as Greedy Growing Convex Front Advancing (GGCFA) and Peeling Field Generation (PFG). Two additional extensions to the GGCF method have also been developed and takes the output of the GGCF method and tries to make improvements to it in different ways. Figure 4.1 shows an overview of the proposed layer generation methods.

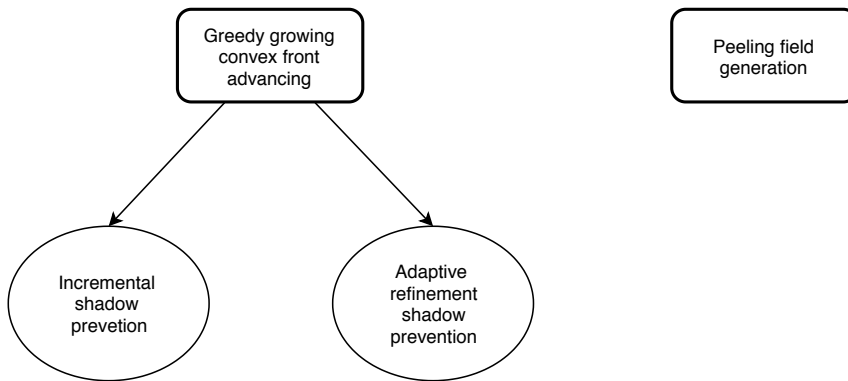


Figure 4.1: Methods for dividing a general object into a sequence of manufacturing layers proposed in Dai et al. (2018) [3].

4.2 Processing pipeline

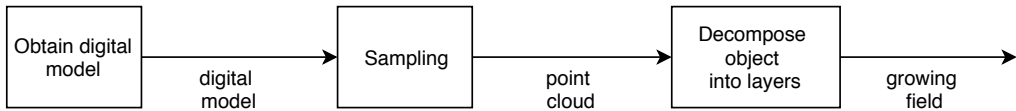
From Section 2.1.5 we know that the basic tasks of many AM programs are:

1. Obtain a digital model of an object

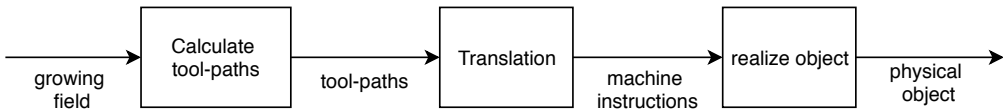
2. Decompose the model into a sequence of layers
3. Calculate tool-paths from the points contained in each layer
4. Translate the tool-paths into machine-readable instructions

How these tasks are implemented and what specific AM technology the program is intended for vary greatly.

The research team in [3] propose algorithms that implement a complete AM program on a 6 DOF industrial manipulator based system. We will now have a look at how these tasks are solved by the framework proposed in the paper for the manufacturing of free-form objects without the use of support structures.



(a) Part 1 of the process – Obtaining an object and decomposing it into manufacturing layers.



(b) Part 2 of the process – Calculating tool-paths from on the manufacturing layers and realizing object.

Figure 4.2: Pipeline of the AM process proposed by Dai et al. (2018) [3].

Figure 4.2 show the specific processing pipeline of the robotic AM system of the paper. The steps an object must go through is:

1. Obtain digital model
2. Sampling
3. Decompose object into layers
4. Calculate tool-paths

5. Translation to machine instructions
6. Object realization

The steps will now be reviewed one by one in order to develop an overall understanding of the framework before we focus on the above mentioned layer generation methods specifically.

4.2.1 Obtain digital model

The paper does not state how the objects used in the demonstrations or experiments are obtained. Common methods such as scanning of real-life objects and using CAD software have already been presented in Section 2.1.5 and are still relevant in this context.

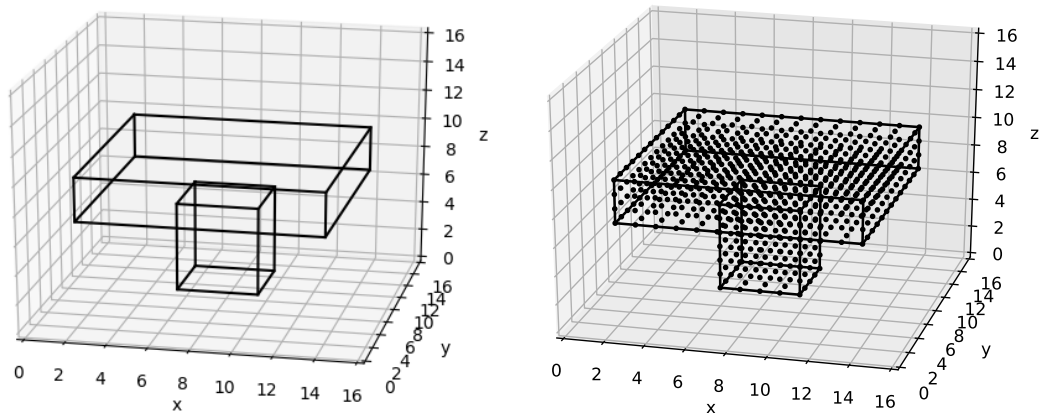
4.2.2 Sampling

The input to the layer generation methods is a set of the coordinates of all the points that the object consist of. A continuous body consists of infinite points in space and if the number is not reduced in some way, the task of searching for a sequence of accumulation layers would be impossible in this case. The process referred to here is known as sampling or discretization of a continuous body. The sampling or discretization process consists of dividing the volume into a final amount of units which we will refer to as voxels. Figure 4.3 shows the sampling process of a 3D object. In Figure 4.3a we see the digital model and in Figure 4.3b we see the same digital model after sampling visualized as a point cloud, i.e. the center point of each voxel is plotted in a graph.

The details of my solution to this problem, and how the models in Figure 4.3 was generated is presented in section 5.1.2.

4.2.3 Accumulate voxels into printing layers

As we know from Section 4.1, the paper present two methods that are used for the generation of manufacturing layers, together with some extensions that aim to improve the sequence of layers suggested by the greedy strategy. The one we will investigate in this thesis is the GGCFA metho focus on, i.e. Algorithm 1 in Appendix B accumulates the voxels in a bottom-to-top approach. The other method accumulates the voxels in the opposite order, i.e. in a top-down approach.



(a) A digital 3D model visualized before sampling.

(b) Sampled version of the object, visualized as a point cloud.

Figure 4.3: Sampling process of a 3D object.

This, together with the tool-path calculation, is the main focus of the paper and detailed descriptions of the different methods and working constraints are given in the paper.

4.2.4 Calculate tool-paths on each layer

After the manufacturing layers have been successfully determined, each layer must be covered with material. The layers can be considered as being 3D surfaces. For each layer, a path connecting all the voxels contained in the surface must be to be calculated. Many algorithms may be used for this task.

Considering position-, orientation- and pose continuity of the tool, the method for calculating continuous tool-paths in the paper is based on Fermat spirals [48] using a geodesic metric. Depending on the complexity of the manufacturing surfaces, any path-finding algorithm with non-intersecting paths may be used for this purpose.

4.2.5 Translate tool-paths to machine-readable instructions

To conclude the pre-manufacture phase we need to feed the tool-paths to some hardware for it to run. In the context of this thesis, the hardware will be a 6DOF robotic manipulator. To do this, the output from the program must be converted

into something that the hardware can read, which is often specific to which type of hardware used.

4.3 Overview of layer generation method

The scope of this thesis will be on one of the methods for solving the decomposition problem from Section 4.1, including one of the method-extensions. The methods we will focus on is highlighted in Figure 4.4, and is the GGCFA method with and without Incremental Shadow Prevention (ISP). The supplementary document corresponding to the paper by Dai et al. (2018) propose algorithms for the implementation of both the method and the extension. In this thesis, the algorithms can be found as Algorithm 3 and Algorithm 4 in Appendix B.

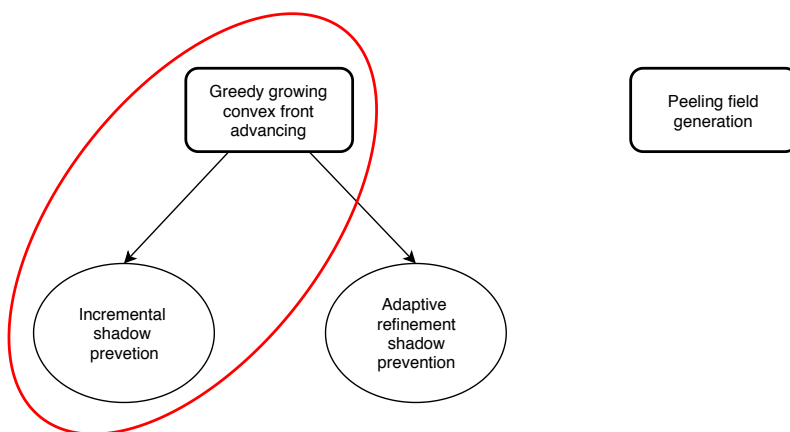


Figure 4.4: Implementation scope of the thesis.

The program implemented by Algorithm 1 will be referred to as Greedy Convex Front Advancing (GCFA) or greedy scheme in this thesis and is a program for dividing an arbitrary object into a sequence of manufacturing layers. Algorithm 2 is a solution improvement scheme, which is a program for finding a better solution based on the solution found by the greedy scheme. This section will give an overview of both methods before we look at them in greater detail later in the chapter.

4.3.1 Greedy scheme

The greedy scheme takes an arbitrary digital model, i.e. an arbitrary digital model of an object, as input. Before the object can be processed by the program, it must be sampled into a 3D grid of separate, equal size, units, referred to as voxels. In the scope of this thesis, a voxel will serve as a two- or three-dimensional point in space.

The GCFA algorithm processes each voxel and accumulates one by one into a specific sequence of manufacturing layers with aim of classifying all voxels of the object into a layer. The sequence of manufacturing layers is generated in a bottom-to-top manner. The algorithm starts by adding all points adjacent to the printing platform into the first layer. The algorithm then searches through every voxel of the object and adds the ones that satisfy some criteria into specific layers. There are no constraints on the shape of the generated manufacturing layers, meaning that they can be curved instead of strictly planar, different from the uniform slicing in Section 2.1.5. A consequence of this is that it needs to be run on a system that can specify both position and orientation, as was investigated in Section 3.2 and Figure 2.9.

4.3.2 Improvement scheme

As we will see later in the thesis, for some objects the greedy approach is not optimal and sometimes omits the accumulation of voxels. Therefore, the research team develops different strategies to improve the result. Algorithm 2 is an extension to Algorithm 1 that evaluates the layer generated by the greedy scheme and searches for a better solution using a one-step look-ahead approach.

4.4 Convex front advancing (CFA)

In this section, the greedy scheme with and without improvement, reviewed in Section 4.3, will be explained and demonstrated. This is the third block of the processing pipeline presented above, see Figure 4.5.

The first method is greedy convex front advancing (GCFA) and the second method is GCFA with incremental shadow prevention (GCFA-ISP). Important steps of the algorithms will be explained in detail together with simulation plots to visually demonstrate how the program works. To implement the program successfully, some changes were made to the original algorithms in Appendix B. The altered algorithms can be found in this section as Algorithm 1 and 2 and the

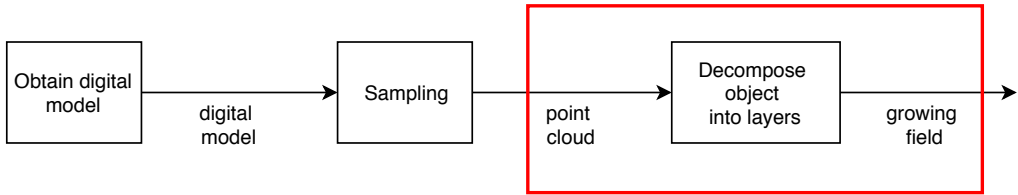


Figure 4.5: The part of the pipeline of the robotic AM system we will study in this section.

changes will be discussed in Section 6.1.

4.4.1 Understanding the results

Section 4.4.3 and 4.4.4 go through the GCFA method and Section 4.4.5 and 4.4.6 go through the GCFA-ISP method. For the sake of demonstration, both programs will be run on the same object so that a comparison easily can be made of the results.

The programs output a sequence of manufacturing layers, referred to as the growing field. A visualized growing field will often be referred to as a result. Each simulation output will show layers generated up to a certain point in time. For consistency in the results, all a layer in the sequence will always be visualized in one color. E.g., layer 1 will, for every object, take the color gray and layer 2 will always take the color purple. Figure 4.6 show an overview of all layer colors of every visualized growing field.

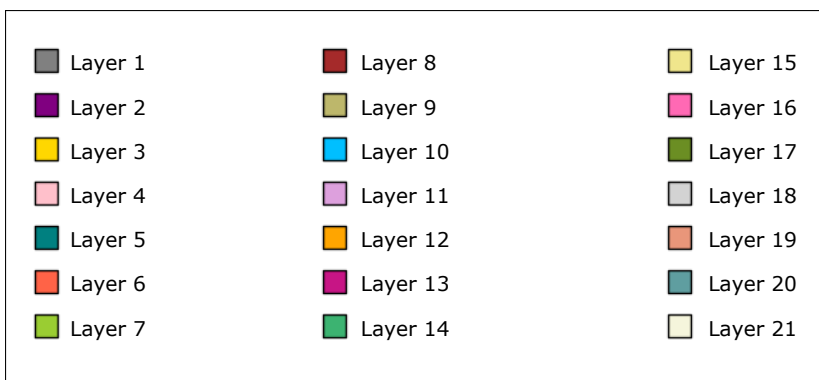


Figure 4.6: Color mapping of layers in growing field.

The plots show position in space, so the axes will represent either x-,y- or z- coordinates in space. The axes will not be named with units, because they represent some undetermined unit in space. This means that the axes can be, e.g., centimeters, millimeters, inches or kilometers. This name of the unit is not relevant for the calculations in this thesis, so it is left out to avoid confusion.

4.4.2 Notation

Summary of the notation used in the algorithms and in the working examples of Section 4.4.4 and Section 4.4.6:

- \mathcal{G} - the growing field
- \mathcal{L} - layer of the growing field
- \mathcal{C} - convex front
- \mathcal{S} - set of shadowed voxels
- \mathcal{T} - working platform

4.4.3 Greedy scheme

Algorithm 1 below implements a greedy scheme for generating manufacturing layers from an input object and is based on algorithm 3 in Appendix B.

Greedy algorithms use a strategy or a problem-solving heuristic of making the locally optimal choice at each stage to try to find a globally optimal solution [49]. They simply choose the option that gives the best result at the point in time of making the decision. In the context of voxel accumulation and printing layers, the greedy CFA approach choose to add a voxel to the next layer if it satisfies some immediate criteria without looking ahead to see the consequences of it being added.

We will now review the specific criteria of the algorithm for guaranteeing manufacturability of the growing field. Each criterion takes form as a constraint-definition pair, i.e. a constraint together with a mathematical definition that implements the associated constraint.

Support-free manufacturing

We start by defining the concept of neighboring voxels. Neighboring voxels are defined as a measurement to determine if a voxel can be manufactured without

Algorithm 1 Greedy convex front advancing

Input: Discrete representation of a solid model, $\bar{\mathcal{H}} = \{v_{i,j,k}\}$ **Output:** A growing field \mathcal{G}

```

1: Add all voxels adjacent to the platform  $\mathcal{T}$  to the first layer  $\mathcal{L}_1$ 
2: Set  $\mathcal{L}_1$  as the current working layer  $\mathcal{L}_c$  and  $\mathcal{C}_{prev} = \emptyset$ 
3: Set the set of already processed voxels  $\mathcal{V} = \emptyset$ 
4: Set the growing field  $\mathcal{G} = \emptyset$ 
5: while  $\mathcal{L}_c \neq \emptyset$  do
6:   Add all voxels of  $\mathcal{L}_c$  into the already processed set,  $\mathcal{V}$ 
7:   Compute the new inaccessible region  $\mathcal{R}$  as  $\mathcal{R} = \mathcal{C}_{prev} \cup \mathcal{L}_c \cup \mathcal{T}$ 
8:   Set the current convex front  $\mathcal{C}_c$  as the convex hull of  $\mathcal{R}$ 
9:   Set  $\mathcal{L}_{next} = \emptyset$ 
10:  for each  $v_{i,j,k} \in \mathcal{L}_c$  do
11:    for each  $v_{r,s,t} \in \mathcal{N}(v_{i,j,k})$  do
12:      if  $v_{r,s,t}$  not inside  $\mathcal{C}_c$  then
13:        if  $v_{r,s,t} \notin \mathcal{V} \cap v_{r,s,t} \notin \mathcal{L}_{next}$  then
14:          Add  $v_{r,s,t}$  into  $\mathcal{L}_{next}$ 
15:        end if
16:      end if
17:    end for
18:  end for
19:  Add  $\mathcal{L}_{next}$  to the growing field  $\mathcal{G}$ 
20:  Set  $\mathcal{L}_c = \mathcal{L}_{next}$  and  $\mathcal{C}_{prev} = \mathcal{C}_c$ 
21: end while
22: return  $\mathcal{G}$ 

```

needing additional support. For an object to be built by material accumulation, new material must always be in contact with *already solidified material*. The paper states that if one of the neighbors of a voxel has previously been manufactured, the voxel itself can be printed without support because it can be attached to the neighboring voxel. The material used must be in a semi-fluid state and fast curing as it will be affected by gravity. This is explicitly stated in Constraint 1.

Constraint 1. (*Support-free*) A voxel can only be accumulated if one of its ASNs has already been solidified.

Constraint 1 is ensured by using the norm of the difference between a voxel

and the surrounding voxels. This concept is referred to as voxels being AM-stable neighbors, see Definition 3. An illustration of the AM stable neighbors of a voxel in two-dimensional space can be found in Figure 4.7. This particular definition differs from the one used in the paper. We will discuss the changes and why they had to be made later in Chapter 6.

Definition 3. Two voxels, $v_{i,j,k}$ and $v_{r,s,t}$, are defined as *AM-stable-neighbours* (ASN) if $\|(i, j, k) - (r, s, t)\|_1 \in \{1, \sqrt{2}\}$

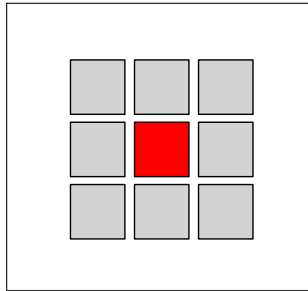


Figure 4.7: AM stable neighbors (ASNs) of the center point, as defined in Definition 3. As long as the center point have been previously manufactured using a rapidly curing material, each ASN(center point) is guaranteed to be manufactured in a support-free manner.

Collision-free manufacturing

The constraint of self-supporting material accumulation is not sufficient to guarantee the manufacturability of each manufacturing layer. We also need to ensure that the printing tool can access each point of the manufacturing paths without colliding with the manufacturing platform or already cured material, see Constraint 2.

Constraint 2. (*Collision-free*) When adding a new voxel to a set of already fabricated voxels \mathcal{V} , the motion of the printer-head should not collide with \mathcal{V} .

A common way to satisfy Constraint 2 is to search for a collision free sequence of voxels to be added to the growing field. In this case, collision detection must be performed on every voxels added. The test objects in the paper consist of 70

000 to 600 000 voxels, so this approach is bound to be very time-consuming. The researcher instead propose an approach for *maintaining* an accessible working surface.

Assuming an arbitrarily shaped printing nozzle, the authors propose to use the convex hull of the already cured voxels, i.e. set of processed voxels \mathcal{V} and the platform \mathcal{T} as a conservative accessible surface. The convex hull of a set of points is the smallest convex set that contains the points [50]. The convex hull takes shape as a convex polyhedron and creates a surface that can be reached independently of the size of the printing tool. A polyhedron consists of n polygonal faces interconnected by $n + 1$ vertices. A convex polyhedron is made from a convex set of points. The mathematical definition of the convex hull can be found in Appendix A.

Definition 4. The *current convex front* is the convex hull of previously processed voxels \mathcal{V} and the working platform \mathcal{T} and is a conservative accessible surface that can be reached by any shape.

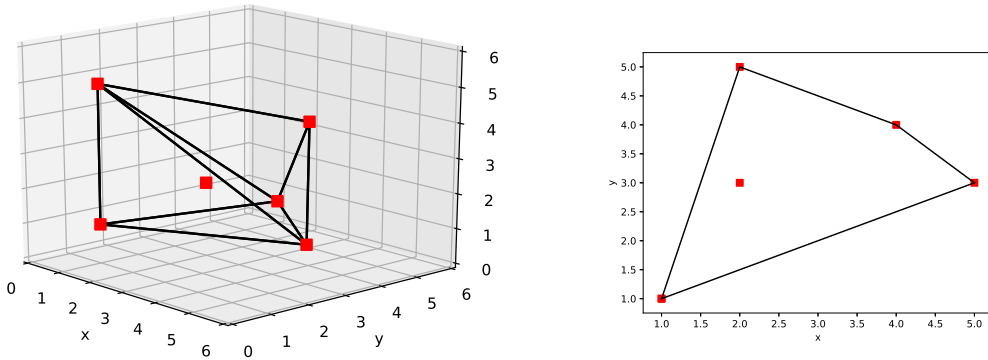
In Figure 4.8 we see illustrations of the convex hull. In three dimensions, the convex hull is a convex polyhedron of a final amount of points [51]. Figure 4.8a shows the convex hull of the set of points $S = (5, 3, 1), (2, 3, 2), (1, 1, 5), (4, 4, 4), (2, 5, 1), (1, 1, 1)$ and Figure 4.8b shows the convex hull of the same points projected onto the x-y plane ($z = 0$).

Manufacturability of every voxel of the growing field

The above constraints and definitions accumulate down to a combined criterion for adding a voxel to the next layer and thereby to the growing field. A point is added to the next layer if it is an AM-stable neighbor *and* if its position is outside the current convex front.

4.4.4 Example: GCFA

This section will demonstrate the details of how the GCFA method works when applied to an object containing a large overhang. The demonstration will hopefully give an interesting result in terms of showing that by dividing the object into curved layers instead of planar, an object that is impossible to realize directly using a 3 DOF system can now be realized directly by utilizing the extended abil-

(a) The convex hull of the set of points \mathcal{S} .

(b) The two dimensional convex hull.

Figure 4.8: The convex hull is a convex polyhedron and can be viewed as a conservative accessible surface.

ities of a 6 DOF system.

The object we will be studying can be seen in Figure 4.9. As we learned in Section 4.2.2, the input to the program is a discretized representation of an object. The result after discretizing the object in Figure 4.9a can be seen in Figure 4.9b. The details of this particular discretization process will be revealed in Section 5.1.2.

The program outputs a growing field \mathcal{G} containing voxels sorted into manufacturing layers and can be applied to both 2D and 3D objects. For this particular example, a 2D object has been chosen because it is easier to visualize the steps of the algorithm in the plane. In Section 5.2 we will look at the result of applying the method to a 3D object.

Determine platform parameters

After we have obtained a discretized version of the object, platform parameters have to be determined. In 2D, only the size (length) of the platform is important while in 3D the size and shape have to be determined. This is an important step because the platform shape have great impact on the output of the program. In Section 5.2 we will see how this affects the layers of the resulting growing field for different object, and a discussion will be made on the topic in Section 6.2.3.

The platform used in this example can be seen in Figure 4.10a. We will refer

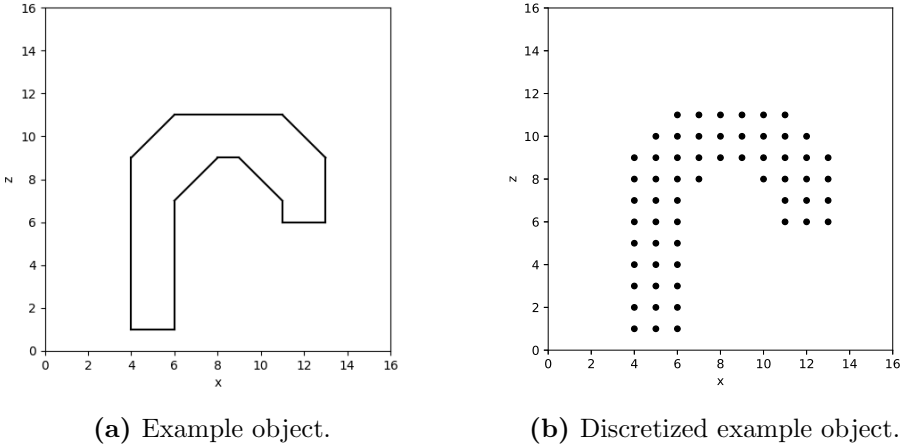


Figure 4.9: This is the example object that will be used in the curved layer generation examples.

to the specified platform as the working platform \mathcal{T} .

Generate initial working layer

The first task of the algorithm is to generate the first layer, \mathcal{L}_1 , of the growing field, i.e. an initial set of voxel coordinates. This is a simple operation of adding all voxels adjacent to the working platform, i.e. all coordinates with z -value equal 1, to a list. The first layer of the example can be seen in Figure 4.10b.

The rest of the layers of the growing field is generated according to the criteria specified in Section 4.4.3 which we will review now.

Finding which voxels to safely add

The purpose of the program is to add qualifying voxels into specific manufacturing layers. Qualifying voxel criteria have been reviewed in Section 4.4.3. To ensure the manufacturability of the growing field, only the voxels that lie inside a safe region can be added to the next layer, \mathcal{L}_{next} . A safe region can be defined as a region in which we can guarantee access by the deposition tool. Each voxel must also be self-supported, i.e. it must be manufactured without the need for additional support structures. Using the terminology of Section 4.4.3, we summarize the three criteria each voxel must satisfy in order to be added to \mathcal{L}_{next} :

1. Must be an AM-stable neighbor (ASN)

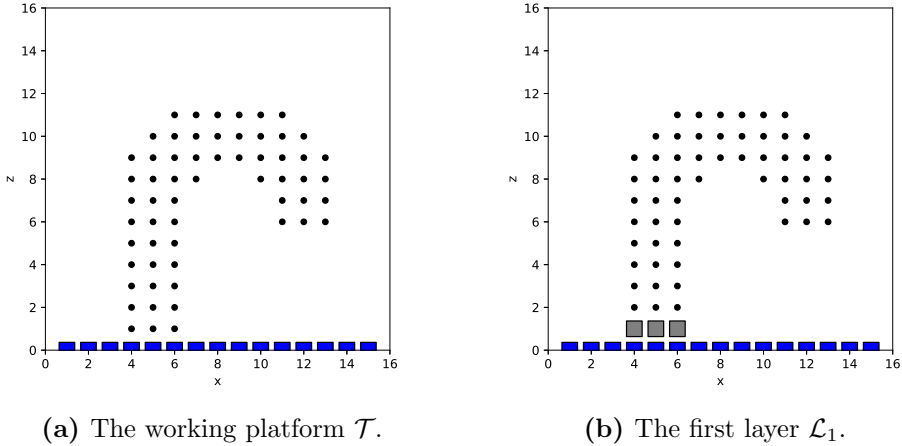


Figure 4.10: The first step of the algorithm is to generate the initial current manufacturing layer $\mathcal{L}_c = \mathcal{L}_1$. All voxels adjacent to the platform is added to \mathcal{L}_1 .

2. Must lie outside the convex hull of the platform and the processed voxels
3. Must not already be in \mathcal{L}_{next}

Calculate convex front

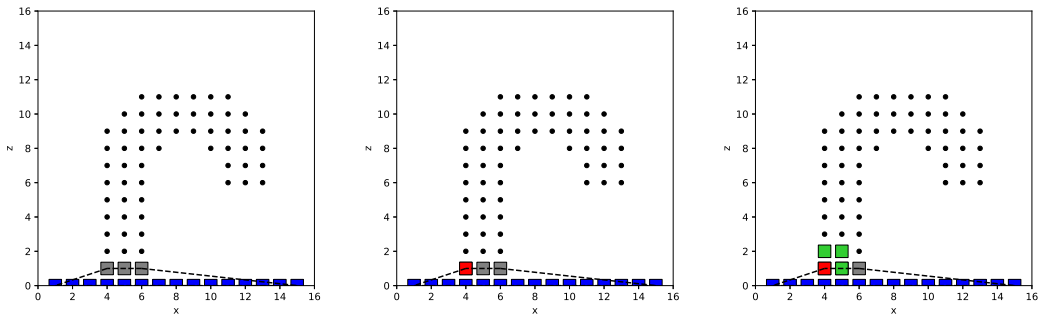
To search for qualifying voxels, the algorithm starts by calculating the current convex front. The current convex front is the convex hull of the union of the previous front, \mathcal{C}_{prev} , the current layer, \mathcal{L}_c , and the platform coordinates, \mathcal{T} , see Equation (4.1).

$$\mathcal{C}_c = \mathcal{C}(\mathcal{C}_{prev} \cup \mathcal{L}_c \cup \mathcal{T}) \quad (4.1)$$

The dotted line in the subfigures of Figure 4.11 show the first calculated convex front, \mathcal{C}_1 , of the program.

Find AM-stable neighbors

Since all voxels that satisfy the criteria of being an ASN have the potential to be added to the next layer the program now all ASNs. Each voxel of the current layer \mathcal{L}_c have a unique set of ANSs. The program starts with the first voxel of \mathcal{L}_1 , visualized in Figure 4.11b, and calculates the ANSs of this voxel. The first set of ANSs can be seen in Figure 4.11c and is found according to Definition 3 in Section 4.4.3.



(a) The dotted line is the first current convex front \mathcal{C}_c of the example. (b) The first voxel of \mathcal{L}_1 is highlighted. (c) The set of ASNs according to the highlighted voxel of \mathcal{L}_1 is found.

Figure 4.11: The process of finding the set of ASNs of a voxel.

Evaluating and adding voxels to the next layer

If an ASN is outside the convex hull and is not already processed and not already in the next layer \mathcal{L}_{next} , it is added to the list that contains the next layer. In the example in Figure 4.11 the voxels that will be added to the next layer is $[4, 2]$ and $[5, 2]$. When the ASNs of the first voxel of the current layer have been evaluated, the algorithm moves on to the next voxel which is $[5, 1]$. The algorithm continues like this until all voxels of \mathcal{L}_c have been evaluated. Figure 4.12 show the resulting second layer \mathcal{L}_2 .

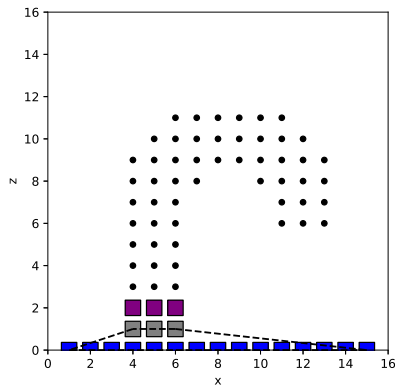
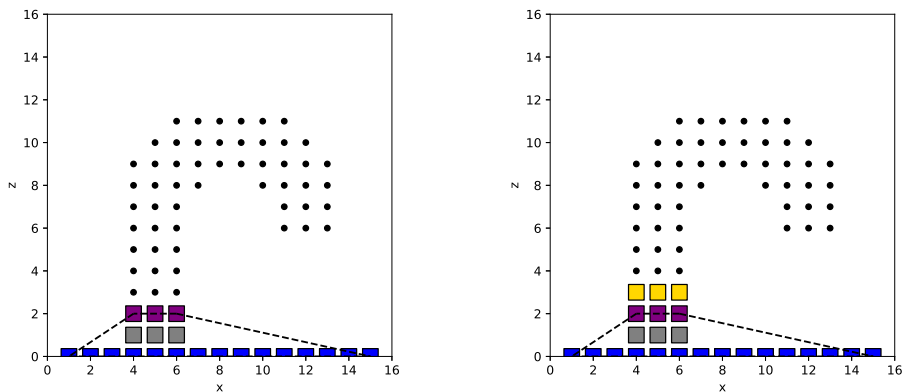


Figure 4.12: Here we see the second layer output by the greedy CFA.

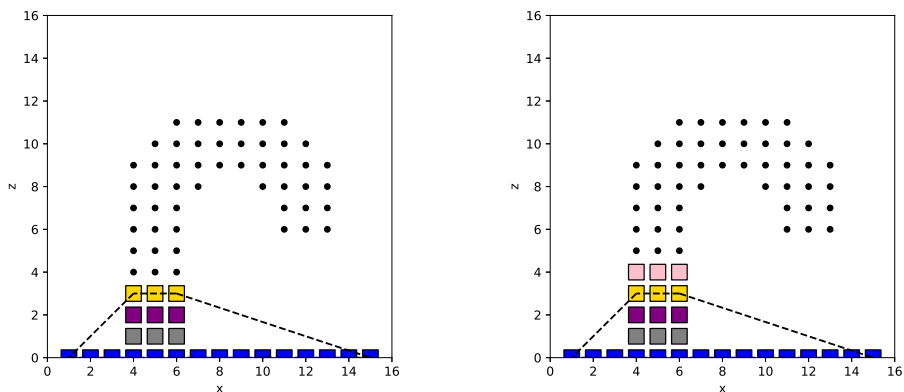
To conclude the while loop-iteration, the current layer is added to the growing field and \mathcal{L}_{next} is made the new current layer. This happens until no new voxels are added to \mathcal{L}_{next} . Figure 4.13 and 4.14 show the generation of layers 3 and 4.



(a) The convex hull of the previous front \mathcal{C}_1 , the platform \mathcal{T} and the current layer \mathcal{L}_2 is calculated as the current front $\mathcal{C}_2 = \mathcal{C}_c$.

(b) The resulting layer $\mathcal{L}_3 = \mathcal{L}_{next}$ generated after considering the ASNs of each voxel of \mathcal{L}_2 .

Figure 4.13: Layer 3 are generated in the same manner as layer 2.



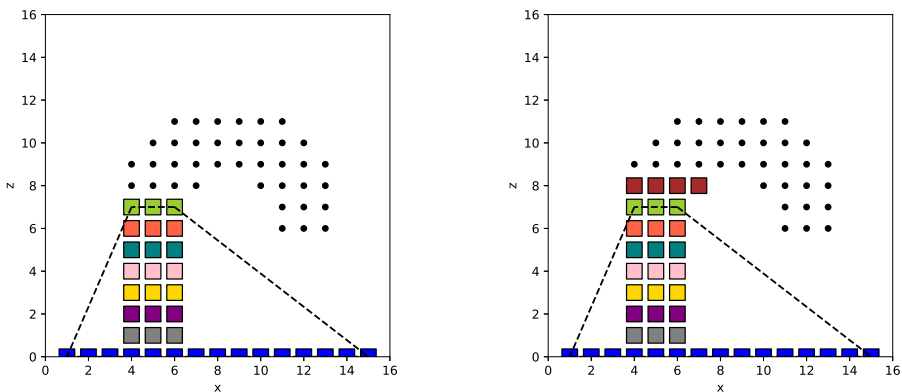
(a) $\mathcal{C}(\mathcal{T}, \mathcal{C}_2, \mathcal{L}_3) = \mathcal{C}_3 = \mathcal{C}_c$

(b) $\mathcal{L}_4 = \mathcal{L}_{next}$

Figure 4.14: Layer 4 are generated in the same manner as layer 3.

Accumulation of voxels in overhangs

In Figure 4.15a we see the generated layers of the growing field up to layer 7 together with the current convex hull. When the next layer is to be generated, we for the first time witness the strength of the GCFA method. Because we, by using a 6 DOF system, are able to change the orientation of the manufacturing tool, the 8. layer which is generated with an overhanging region, can be realized and thus it is shown that the method can be used to directly manufacture objects with overhang.



(a) The growing field up to \mathcal{L}_7 .

(b) The first accumulation of voxels in an overhanging region.

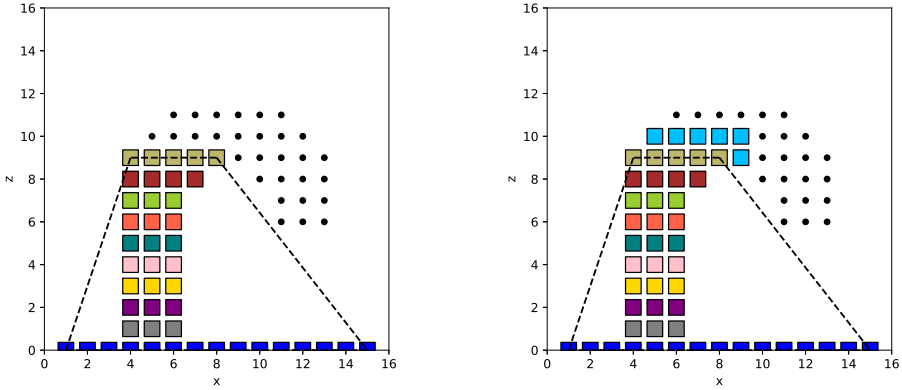
Figure 4.15: Accumulation of voxels in critical region.

Generation of curved manufacturing layers

Figure 4.16a show the generated growing field up to layer 9. All layers up until this point have come out planar. After this we can see in Figure 4.20a that we have the first occurrence of a non-planar, or curved, layer. By accumulating voxels to form curved layers in the growing field affect the ability of the method to successfully decompose objects of even larger overhangs.

Weakness of greedy strategy

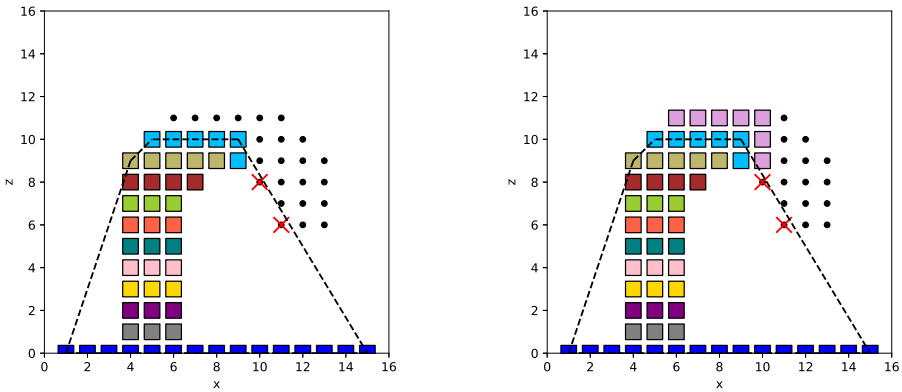
Figure 4.17 demonstrates an important weakness with using a greedy strategy for the accumulation of voxels into manufacturing layers. As explained at the beginning of Section 4.4.3, greedy strategies make the locally optimal choice to try to find a globally optimal solution without considering the consequences of



(a) \mathcal{L}_1 to \mathcal{L}_9 of the growing field are planar layers. (b) \mathcal{L}_{10} is the first curved layer of the growing field

Figure 4.16: The first occurrence of a curved layer, \mathcal{L}_{10} .

the decision.



(a) \mathcal{C}_c is calculated before the 11. is to be generated. (b) Generation of the 11th layer cause two voxels to be shadowed.

Figure 4.17: The first occurrence of shadowed voxels. The red crosses show voxels that cannot be added to the next layer due to being inside the current convex front. The algorithm have no means to prevent voxels from being shadowed.

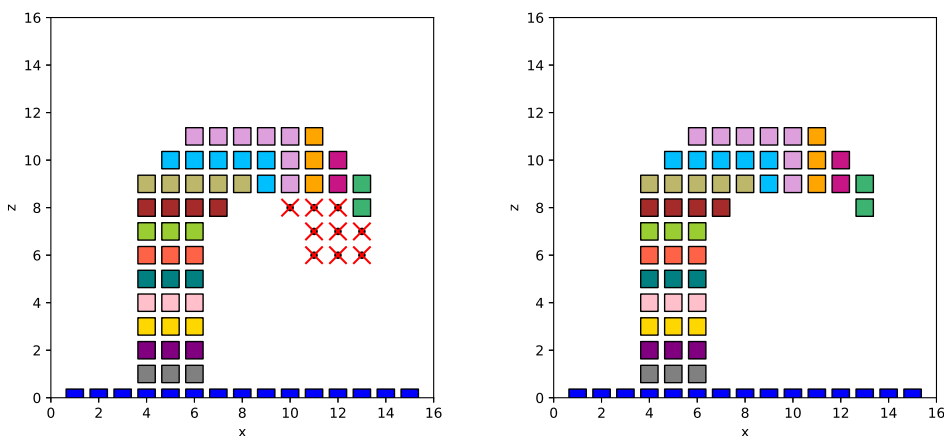
In Figure 4.17a we see the ten first layers together with the current convex front. We see that two points of the object are placed inside the convex front, i.e.

the red x-es in the figure. This means that independent of whether the voxels are ASNs or not, these voxels can not be added to the next layer at this iteration. A property of the convex front is that it can never be reduced in size, so once a voxel is inside, it will stay inside for the remaining part of the program. This again means that the two concerned voxels can never be added to any future layer of the growing field.

Figure 4.17b show the resulting 11. layer together with the omitted voxels of the object.

The resulting growing field

The growing field output from the GCFA method, i.e. the resulting sequence of manufacturing layers, after traversing through every voxel of the example object can be seen in Figure 4.18. We clearly see the generation of curved layers that enable for the manufacturing of large overhangs. The method does, however, not give a perfect result, i.e. some voxels are omitted from the solution. This is because the greedy scheme could not find a way to include these voxels in the manufacturing layers. Next, a method for improving this result will be reviewed and demonstrated.



(a) The growing field together with the omitted voxels. (b) The growing field visualized alone.

Figure 4.18: The resulting growing field \mathcal{G} of the GCFA example

4.4.5 Improvement scheme

Algorithm 2 below implements the incremental shadow prevention (ISP) program, which is an extension added to the greedy scheme (GCFA) to improve the layers of the growing field \mathcal{G} in terms of omitting fewer voxels, based on Algorithm 4 in Appendix B. The altered algorithm has some minor changes compared to the original one, which will be further discussed in Chapter 6.

We witnessed in the demonstration of the GCFA program that the result, Figure 4.18, ended up with omitting a relatively large amount of voxels. This was due to the property that the convex front was never reduced in size, so once a voxel was inside the front it could never be outside again.

The ISP extension evaluates if a layer is the best choice before it is added to the growing field. If it finds a better solution, this solution is made the new next layer. Before explaining how this is done we need to be familiar with some new terminology defined in the paper.

Definition 5. A voxel $v_{i,j,k}$ is shadowed if it is unprocessed but lies inside the convex hull of the current advancing front \mathcal{C}_c . A set of shadowed voxels form a shadow region.

Constraint 3. (*Shadow-prevention*) When adding new voxels onto a set of already fabricated voxels \mathcal{V} , the number of shadowed voxels should increase as little as possible.

The incremental shadow prevention method computes new alternative convex hulls so that no voxels are shadowed unnecessary. It takes the next layer \mathcal{L}_{next} determined by the greedy strategy and searches for alternative layers that generate less shadowed voxels, resulting in a, hopefully, smaller shadow region in total. We say hopefully here because, as we will see later, the method only looks one step ahead before making a decision and therefore is not guaranteed to find a globally better result.

Algorithm 2 Incremental scheme for shadow prevention

Input: The model $\bar{\mathcal{H}} = \{v_{i,j,k}\}$, the platform \mathcal{T} , the set of processed voxels \mathcal{V} , the next layer \mathcal{L}_{next} , a convex front \mathcal{C} , the set of voxels shadowed by the corresponding convex front \mathcal{S}

Output: A reduced set of \mathcal{L}_{next}

- 1: Set the reduced next layer $\tilde{\mathcal{L}}_{next} = \emptyset$
 - 2: Set the new processed set of voxels $\tilde{\mathcal{V}} = \emptyset$
 - 3: Compute $\tilde{\mathcal{C}}$ as the convex hull of $\mathcal{C} \cup \mathcal{T} \cup \mathcal{L}_{next}$
 - 4: Add all voxels of \mathcal{V} and \mathcal{L}_{next} into $\tilde{\mathcal{V}}$
 - 5: Add all voxels shadowed by $\tilde{\mathcal{C}}$ into the potentially shadowed set \mathcal{S}_p
 - 6: **if** \mathcal{S}_p contains more voxels than \mathcal{S} **then return** \mathcal{L}_{next}
 - 7: **else**
 - 8: Copy the values of \mathcal{L}_{next} into \mathcal{Q}
 - 9: Set $\mathcal{V}_t = \emptyset$
 - 10: **while** $\mathcal{Q} \neq \emptyset$ **do**
 - 11: Let v be the first element element of \mathcal{Q}
 - 12: Remove v from \mathcal{Q}
 - 13: Compute the new inaccessible region $\tilde{\mathcal{R}} = v \cup \mathcal{T} \cup \tilde{\mathcal{L}}_{next} \cup \mathcal{C}$
 - 14: Compute a test convex front \mathcal{C}_t as the convex hull of $\tilde{\mathcal{R}}$
 - 15: **for** $v_{i,j,k}$ in \mathcal{V} **do**
 - 16: **if** $v_{i,j,k}$ not in \mathcal{V} **then**
 - 17: Add $v_{i,j,k}$ to \mathcal{V}
 - 18: **end if**
 - 19: **end for**
 - 20: Add v to \mathcal{V}_t
 - 21: Add voxels shadowed by \mathcal{C}_t to \mathcal{S}_t
 - 22: **if** \mathcal{S}_t contains less voxels than \mathcal{S}_p **then**
 - 23: Add v to $\tilde{\mathcal{L}}_{next}$
 - 24: **end if**
 - 25: **end while**
 - 26: **if** $\tilde{\mathcal{L}}_{next} \neq \emptyset$ **then**
 - 27: Set $\mathcal{L}_{next} = \tilde{\mathcal{L}}_{next}$
 - 28: **end if**
 - 29: **return** \mathcal{L}_{next}
-

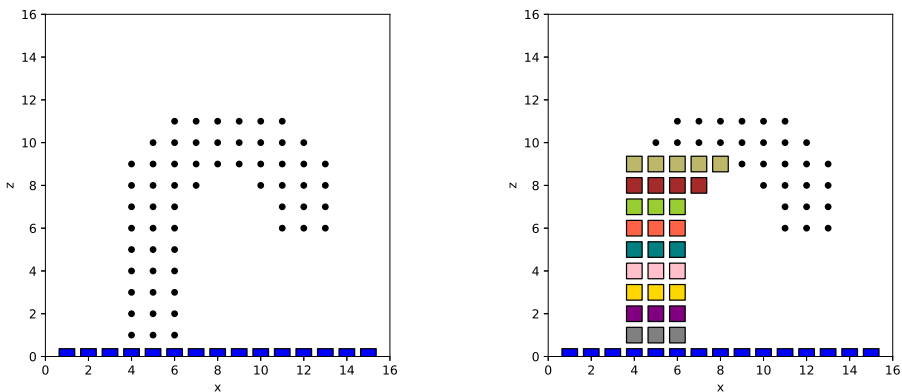
4.4.6 Example: GCFA-ISP

As stated in Section 4.4.5, the incremental shadow prevention approach computes new alternative convex hulls so that no voxels are shadowed unnecessarily. See Section 4.4.5 for the definitions of shadowed voxels and – regions. If the method finds a better solution, this solution is returned as \mathcal{L}_{next} and becomes the input to the next iteration of the algorithm.

In this section, it will be demonstrated how this happens. We use the same approach as in Section 4.4.4, i.e. showing outputs at interesting steps to visually demonstrate what happens during the program. We also use the same object and working platform, see Figure 4.19a, so that the results can be easily compared and evaluated.

Layers that do not generate shadowed voxels

Figure 4.19b shows layer 1 to layer 9 from the growing field for both *both methods*. These layers are equal for both methods because no voxels are shadowed by the greedy method up until this point.



(a) The input object together with the working platform.

(b) Layers 1-9 are the same for both methods.

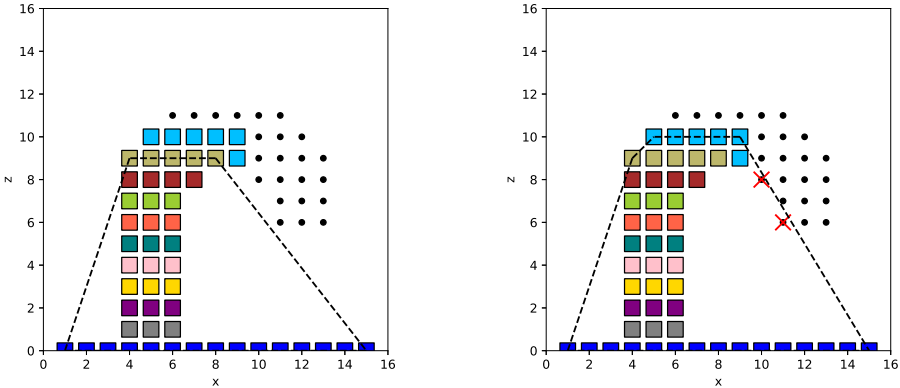
Figure 4.19: We use the same example object for the GCFA-ISP example as in the GCFA example in Section 4.4.4. As long as no voxels are shadowed by the sequence of convex fronts, the ISP improvement algorithm do not interfere with the greedy output.

Triggering of the ISP method

Figure 4.20 show the sequence of operations that result in the first occurrence of shadowed voxels of the example. The ISP method will look at the consequence of adding the current layer to the growing field in terms of potential shadowed voxels. Since it looks one step ahead, a layer that is proposed by GCFA and contributes to generate shadowed voxels when used to generate the *next layer* will trigger the method.

In Figure 4.20a we see layer 1 to layer 9, the convex front of the platform, the previous front and layer 9, together with the new greedy generated layer 10. In Figure 4.20b we see what happens in terms of shadowed voxels when adding this particular 10th layer to the growing field. What we see is that this action directly causes the two marked voxels to be shadowed.

When this happens, the ISP program will try to remove from layer 10 the voxels that directly cause the shadowing of these two voxels.



(a) \mathcal{L}_{10} as proposed by GCFA.

(b) Consequence of choosing this \mathcal{L}_{10} .

Figure 4.20: If \mathcal{L}_{10} is chosen as it is here, two voxels will be shadowed when determining the next layer by the greedy approach. An attempt should be made to "save" these voxels before continuing the program.

Preventing shadowing of voxels

We will now go through how the ISP proceed in order to try to reduce the number of voxels shadowed by the next layer, i.e. the set of potential shadowed voxels.

The layer in Figure 4.20a that is objective for improvement consist of the points:

$$\mathcal{L}_{10} = [[5, 10], [6, 10], [7, 10], [8, 10], [9, 10], [9, 9]]$$

And the potential shadow region we want to reduce is:

$$\mathcal{S}_p = [[10, 8], [11, 6]]$$

With aim of reducing the potential shadow region, the ISP algorithm chooses coordinates from \mathcal{L}_{10} and checks which ones produce all or parts of the shadowed region.

Choosing to start with the first element of \mathcal{L}_{10} , we create a new list with all the elements of \mathcal{L}_{10} and removes the first element. This is what happens in the algorithm:

$$q = [[5, 10], [6, 10], [7, 10], [8, 10], [9, 10], [9, 9]]$$

1. *iteration:*

$$p = [5, 10]$$

$$q = [[6, 10], [7, 10], [8, 10], [9, 10], [9, 9]]$$

1. $\mathbf{p} = [5, 10]$

To check if $p = [5, 10]$ produces any additional shadowed voxels we compute the convex hull in Equation (4.2).

$$\mathcal{C}(p, \text{platform}, \text{reduced_next_layer}, \text{previous_front}) \quad (4.2)$$

Since it is the 1. iteration, no voxels have been added to the reduced next layer yet, so this set is empty. Figure 4.21a shows the convex hull from Eq. 4.2. We see that adding p to the next layer does not increase the amount of shadowed voxels (it is still zero), so it is added to the reduced next layer.

2. $\mathbf{p} = [6, 10]$

Since q contains more points, we continue by setting $p = [6, 10]$ and computes a new convex hull. Figure 4.21b shows the convex hull from the second iteration with $p = [6, 10]$. We see that no new voxels have been shadowed so this point is also added to the reduced next layer.

3. $\mathbf{p} = [7, 10]$

See Figure 4.21c. This point do not shadow any voxels. $p = [7, 10]$ is added to the next layer.

4. $p = [8, 10]$

See Figure 4.21d. This point do not shadow any voxels. $p = [8, 10]$ is added to the next layer.

5. $p = [9, 10]$

In Figure 4.21e we see that if we were to add $p = [9, 10]$ to the next layer we would cause three voxels to be shadowed. This is more than what we started with, which was two voxels, so this is definitely not an improvement in terms of reducing the shadowed voxel-count. As a result, this voxel is not added to the reduced next layer.

6. $p = [9, 9]$

The last point contained in q is $[9, 9]$. When added to the next layer it produces one shadowed voxel, see Figure 4.21f. This is an improvement in comparison to the greedy next layer which had two shadowed voxels, so $p = [9, 9]$ is added to the reduced next layer.

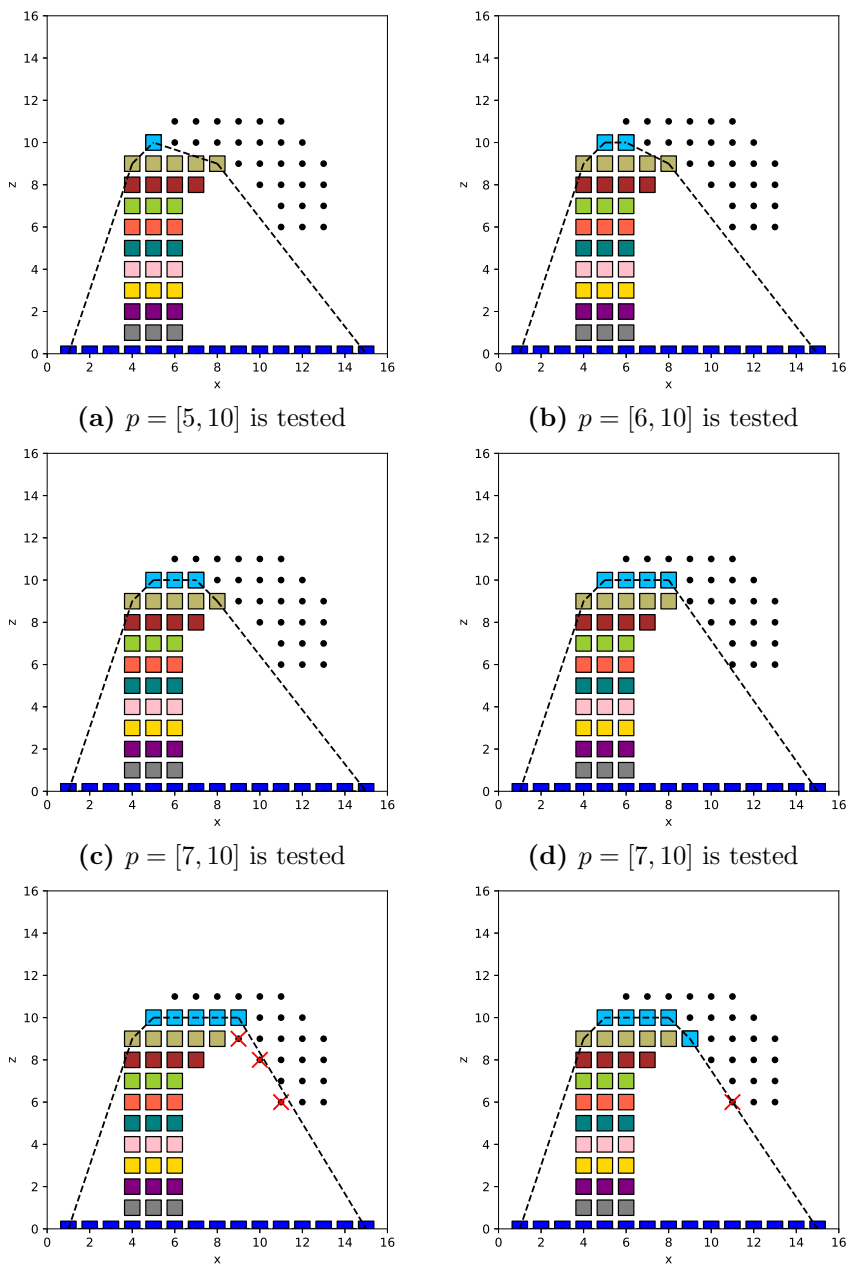
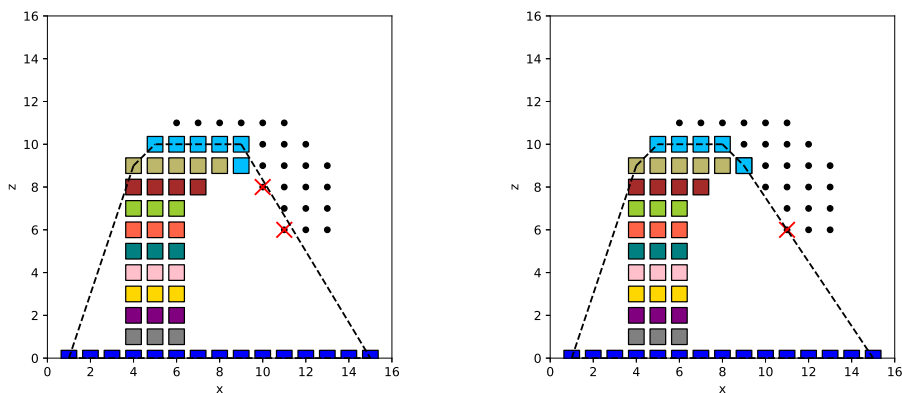


Figure 4.21: Improving the layer proposed by the greedy scheme using ISP.

The resulting improved layer

The resulting next layer generated with incremental shadow prevention contains one less voxel than the pure greedy one and shadows one voxel instead of two. Figure 6.6b



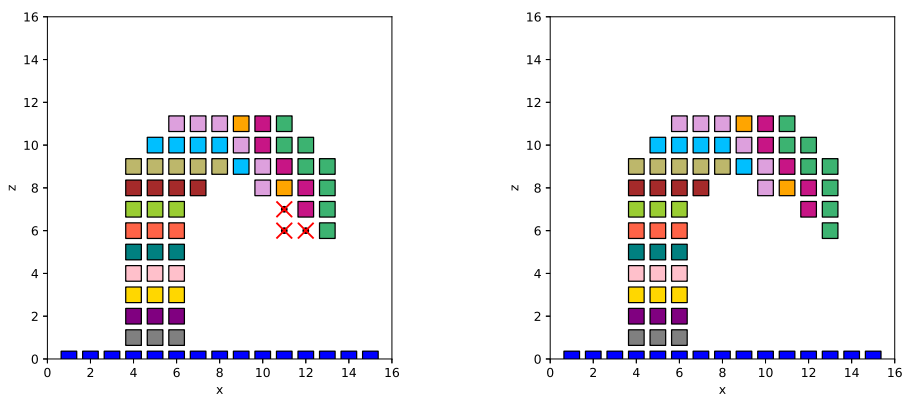
(a) Layer 10 proposed by greedy method.

(b) Layer 10 proposed by greedy method with incremental shadow prevention-

Figure 4.22: comparing the generation of the 10th layer by the greedy approach and by the greedy approach with improvement.

The resulting growing field

After the reduction process of layer 10, the greedy generation of the next layer, layer 11, starts. If this layer do not cause the shadowing of any new voxels, the greedy choice is accepted and the 12th can be generated. Else, it must also be examined by the ISP scheme. The resulting growing field generated by the GCFA-ISP method can be seen in Figure 4.23. We see that the amount of shadowed voxels now is 3, which is a reduction of 6 shadowed voxels compared to using the GCFA method.

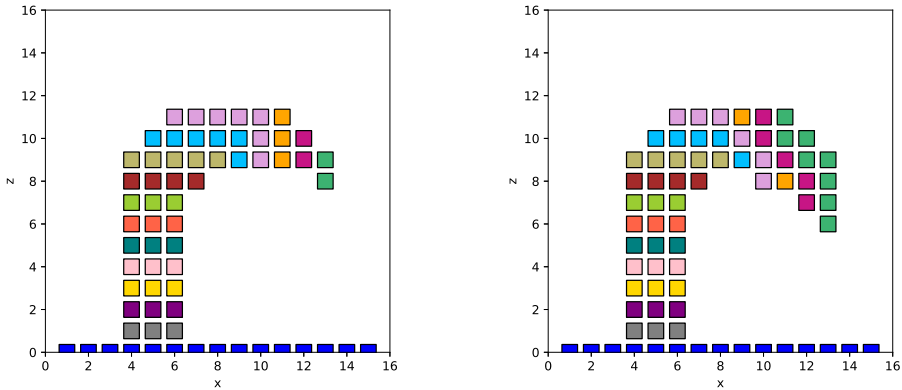


(a) The growing field together with the omitted voxels. (b) The growing field visualized alone.

Figure 4.23: The resulting growing field \mathcal{G} of the GCFA-ISP example.

Comparing GCFA and GCFA-ISP for this example

We see from Figure 4.24 that the solution has been improved in terms of shadowed voxel-count. In Section 5.2 a comparison of the two solutions will be performed. We will now have a look at how this solution is generated.



(a) Growing field \mathcal{G} generated using GCFA.

(b) Growing field \mathcal{G} generated using GCFA-ISP.

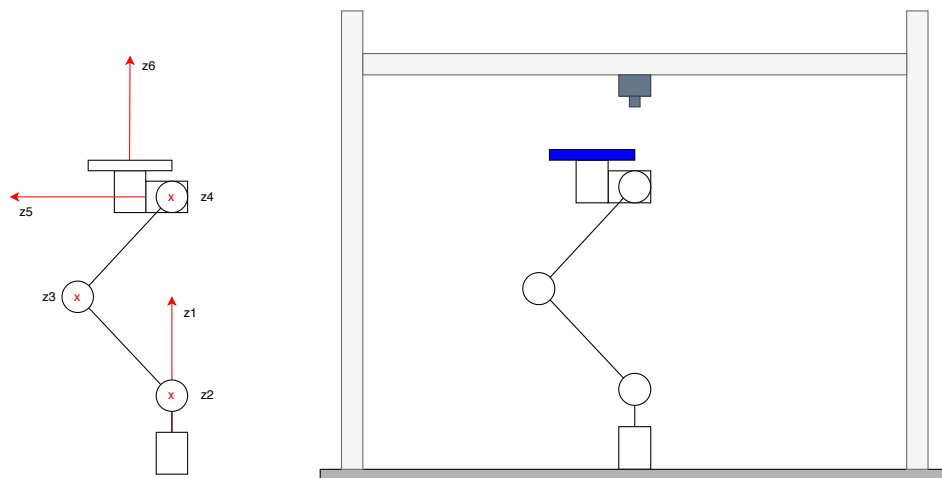
Figure 4.24: Comparing the resulting growing fields.

4.5 Hardware

In this section we will look at the hardware system used in the paper by Dai et al. (2018). This hardware was also mentioned in the literature review of robotic AM methods in Section 3.2.3.

The paper document physical experiments on six different input models. Each input model is some kind of free-form object consisting of various combinations of critical regions such as holes, islands and overhangs. All test objects of the experiment have been manufactured using the same hardware. The system consist of a 6 DOF robot manipulator as defined in Section 2.2 together with a circular platform which is attached to the wrist of the robot arm. Since the robot arm is holding the platform, the deposition tool must be attached elsewhere. The research team choose to attach it to an overhead system in a fixed position, i.e. its position cannot change during fabrication.

In Figure 4.25 we see schematic drawings of the hardware. In Figure 4.25a we see the robot manipulator together with all 6 axes of rotation. The complete hardware setup can be seen in Figure 4.25b. In both parts of Figure 4.25 we see the manufacturing platform which the object will be built at attached to the end of the robot arm. Most developers of AM systems choose to attach the deposition tool to the movable part of the system, whether the system being a gantry or a robot, so the proposed system do stand out in research because of this configuration. In Chapter 6 we will look at some benefits of using a system like this.



(a) The rotational axes ($z_1 - z_6$) of this 6 DOF robot manipulator

(b) The robotic AM system proposed in [3]. A 6 DOF manipulator with the manufacturing platform attached at the wrist.

Figure 4.25: Schematic drawings of the hardware used in [3].

Chapter 5

Implementation and results

In this chapter, details of the implementation of the algorithms from Chapter 4 is presented followed by the result from repeated simulations on different test objects.

5.1 Implementation

Algorithm 1 from Section 4.4.3 and Algorithm 2 from Section 4.4.5 is implemented in full by the author as a part of the work on this thesis. The programs were made with the goal of developing a thorough understanding of the methods and to obtain simulation results, which are presented in Section 5.2. The algorithms have been implemented in the Python programming language, using version 3.6. The third-party open-source libraries used will be presented below. Python built-in modules have also been used. The Pycharm Integrated Development Environment (IDE) has been chosen as the framework of implementation.

See Appendix C for more information on where to find the implementation and details of the modules of the programs.

5.1.1 Dependencies

The implemented program uses some external libraries which are all open-source, meaning that the source code is made freely available for redistribution and modification. In this section, all the dependencies of the program will be reviewed.

SciPy

SciPy is open-source software for mathematics, science, and engineering for use with Python. From SciPy, the implementation uses `scipy.spatial` that implements spatial algorithms and data structures in Python, for calculating the convex hull and for checking whether a voxel is inside the convex hull.

- **`scipy.spatial.ConvexHull`**

The module implements the computation of the convex hull in n dimensions using an external Qhull library [52]. The Qhull library uses the Quickhull algorithm for computing the convex hull [50].

Code 5.1 show how the module is used in the program. It takes an array of the points in which the convex hull should be computed from as parameter and returns the convex hull of the given points as a list of the vertices.

```
def convex_front(region):
    points = np.asarray(region)
    hull = ConvexHull(points)
    front = []
    for index in hull.vertices:
        front.append(region[index])
    return front, hull
```

Code 5.1: Usage of SciPy-function for computing the convex hull of a set of points.

- **`scipy.spatial.Delaunay`**

Computes the Delaunay tessellation in N dimensions. Code 5.2 is borrowed from StackOverflow, an online community for developers [53], and shows how the Delaunay method is used to evaluate if a voxel is inside the convex hull or not in the program.

Matplotlib

Matplotlib is a library that implements plotting methods for python. It has been used to generate all the plots of this work.

```
def in_hull(p, hull):  
    from scipy.spatial import Delaunay  
    if not isinstance(hull, Delaunay):  
        hull = Delaunay(hull)  
    return hull.find_simplex(p) >= 0
```

Code 5.2: Usage of SciPy-function for computing the convex hull of a set of points.

5.1.2 Pre-processing of object

The input to the GCFA and GCFA-ISP program is a set of all the voxels that the object consist of. So before we can start computing, the continuous body we want to realize by AM must be represented in discrete voxel-coordinates.

The most common way to solve this task is to create a digital model using some existing CAD software and transform it into a point cloud, i.e. a set of body-coordinates. Since the background of the author of the thesis is not graphics or anything graphics-related, getting acquainted with a new discipline would require some time. Therefore, the task have been simplified to creating a discrete body direct using plain text-files and binary numbers. This method can be applied in creating 2D objects as well as 3D objects, so it was determined to be sufficient for the scope of this thesis.

In Figure 5.1a we see the shape representation of a 2D object. This object can be translated into a binary representation, see Figure 5.1b. The object is represented by 1-digits and the printing platform by hyphen-signs(-), stored in a plain text-file, i.e. a file with .txt extension. The program, see Code 5.3, reads the binary object using built-in python functions by looping through the characters and stores the coordinates of every point that contains a "-" or a "1" into platform- and object lists.

In 3D, the process of making an object using binary digits is more complicated and time-consuming. The task is solved by creating cross-section layers to bed stacked on top of each other. Figure 5.2 shows a pyramid divided into binary layers together with its platform. For objects that can be sampled using large steps without losing too much information, this method is sufficient and not very time-consuming to perform.

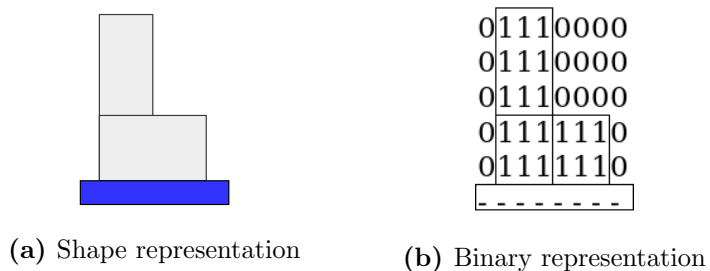


Figure 5.1: Two different ways of representing a 2D object.

```
def load_file_2D(filename):
    f = open(filename)
    for row in f:
        for char in row:
            if char == "1":
                model.append(coordinate)
            if char == "-":
                platform.append(coordinate)
    f.close()
    return model, platform
```

Code 5.3: Loading object and platform from text-file into python lists.

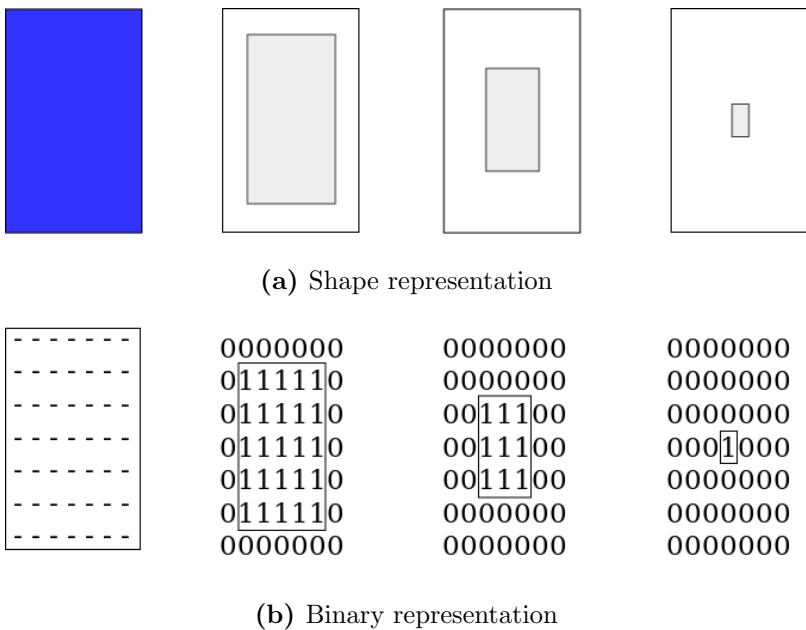


Figure 5.2: Two different ways of representing a 3D object. For simplicity, the object is decomposed into $n = 4$ two dimensional sub-objects.

5.1.3 Convex front advancing

Algorithm 1 and Algorithm 2 have been implemented in its entirety by the author of this thesis. See Appendix C for details on where to find the code. This section will give specific information on the implementations.

Figure 5.3 show the a schematic overview of the interface, i.e. the inputs and outputs, of the implemented program `convex_front_advancing()`. The input to the program is:

- `method` – the method the program should use, can be "greedy" or "isp"
- `nd` – the dimension of the object, can be 2 or 3
- `filename` – the name of the .txt-file that contains the binary representation of the object, e.g. "object1"
- `filepath` – the path to the folder containing the object-file

The block in Figure 5.3 can be viewed as a "black box" computing the growing field, and outputs the growing field together with a list of any shadowed voxels. The details of the implementation of the "black box" for both methods will revealed below.

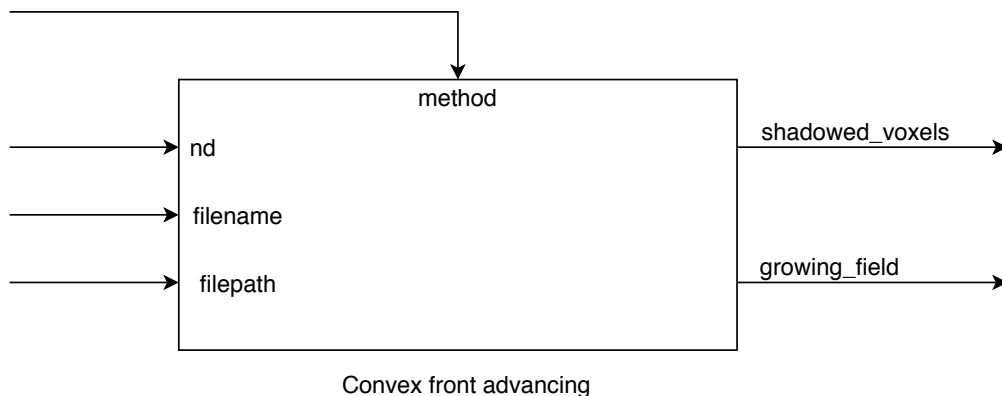


Figure 5.3: Interface of the implemented CFA program.

GCFA

If `method="greedy"` the greedy convex front advancing-program is run. A information-flow overview of what happens inside the above mentioned "black box" in Figure

5.3 in this case can be found in Figure 5.4.

We see that the object specified by `filename` and `filepath` is loaded into the variables `object` and `platform` as explained in Section 5.1.2. The block "Greedy scheme" in this figure computes the layers according what is specified in Algorithm 1. For each layer that is generated in the block, the layer is added to the `growing_field` list, using python built in functions. The same layer is also fed back into the block for the next greedy-iteration of generating the next layer.

GCFA-ISP

If `method="isp"` the method in Figure 5.5 is run. If we compare the method of Figure 5.4 to the method in Figure 5.5 we see that an improvement-block has been added. The improvement-block implements the incremental shadow prevention method specified in Algorithm 2. It takes the `next_layer` list generated by the greedy-block and outputs `reduced_next_layer` which has been improved if improved if possible.

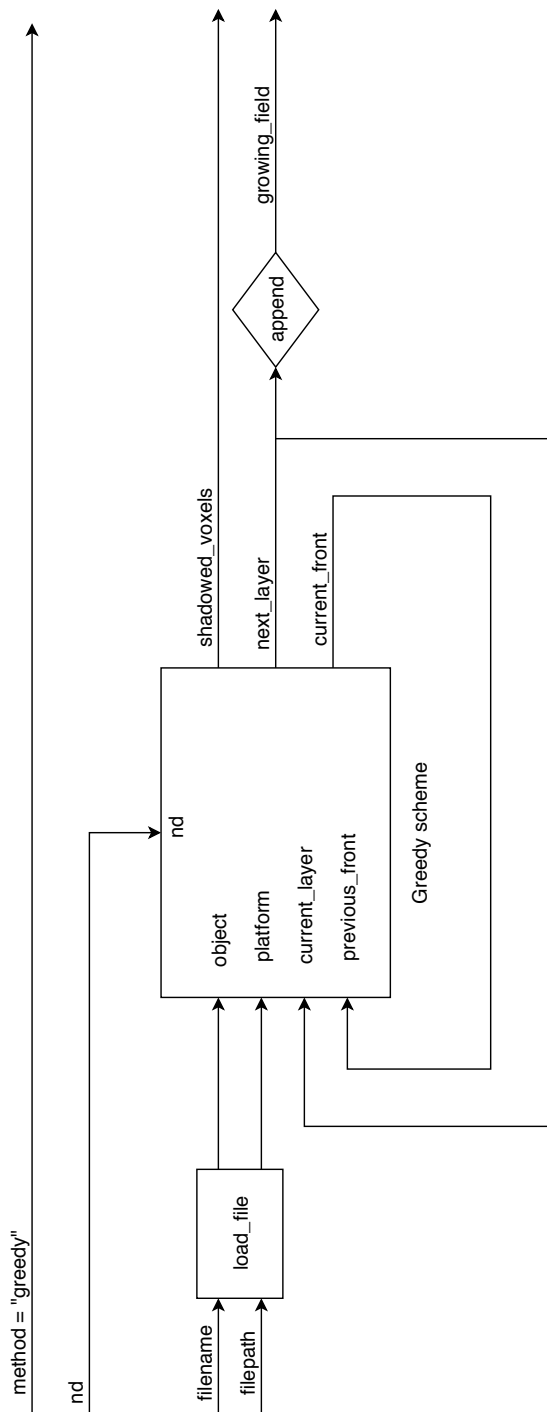


Figure 5.4: Information flow of the GCF A-implementation

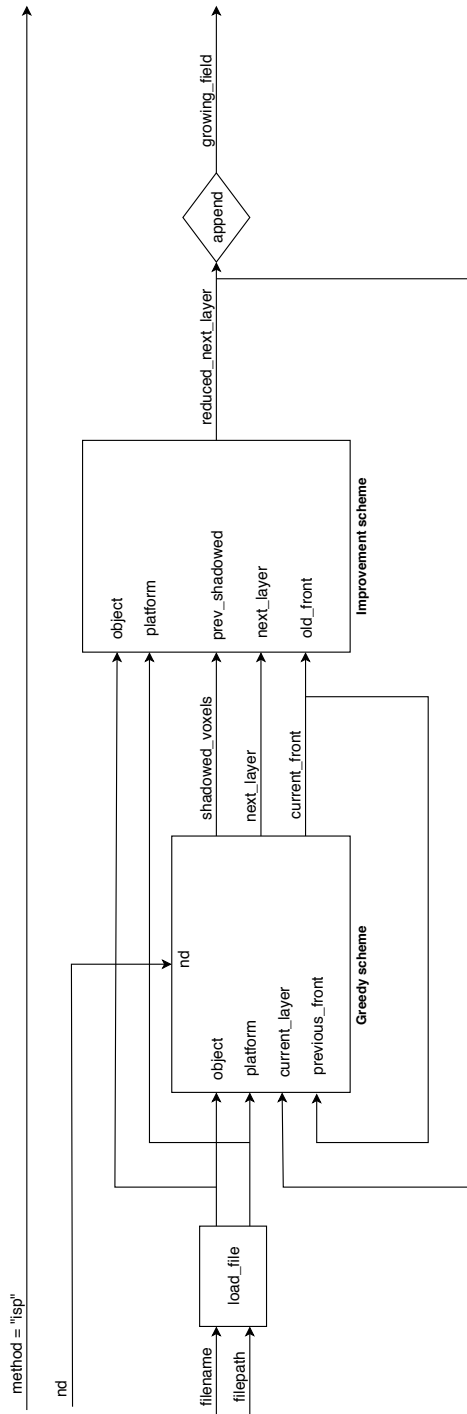


Figure 5.5: Information flow of the GCFA-ISP-implementation

5.2 Results

The program for which the implementation details was presented in Section 5.1.3, Greedy Convex Front Advancing (GCFA) and GCFA with Incremental Shadow Prevention (GCFA-ISP), have been tested on six different input objects. In this section, the chosen test objects will be introduced, followed by a presentation of all simulation results obtained by running the programs. The section is concluded with a summary of all the results. The results will be further discussed in Chapter 6, Section 6.2.

5.2.1 Test objects

In Figure 5.6 we see an overview of the test objects. The objects contain different combinations of critical regions, making each of them interesting for testing.

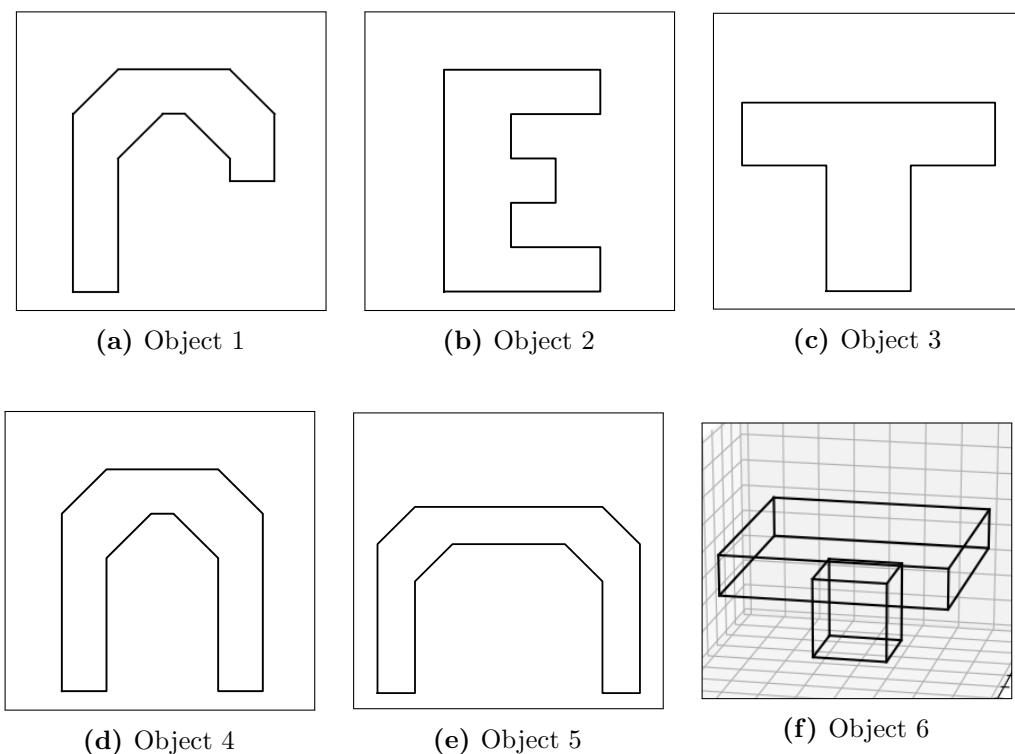


Figure 5.6: Objects used for testing the algorithms from Chapter 4

The test objects have been chosen because traditional AM systems, reviewed in Section 2.1, would not be able to manufacture them without adding support structures or dividing them into pieces for manufacturing.

5.2.2 Presenting the results

We run the GCFA and GCFA-ISP programs on the test object with respect to three different platform sizes. The platforms have been divided into three size groups:

- Small : a platform that is just as large as the object itself.
- Medium : a platform that is slightly larger than the object
- Large : a platform that is much larger than the object

Each layer of the resulting growing field will be represented by colors. The colors used will, as in the demonstrations above, be the ones represented in Figure 4.6. For each resulting growing field, the number of manufacturing layers and the number of shadowed voxels is counted.

Object 1

The first test object is the same object as the one used in the example calculations in Sections 4.4.4 and 4.4.6. The shape-, binary- and point cloud representation of the object is shown in Figure 5.7.

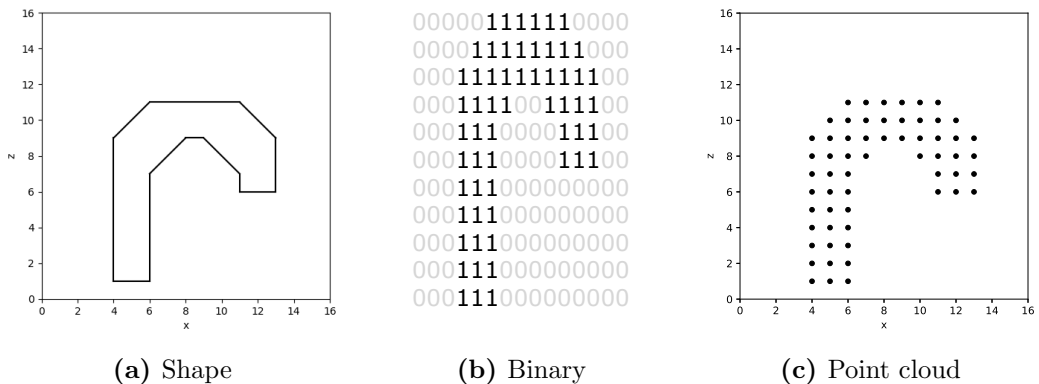


Figure 5.7: Test object 1 represented in three ways.

The first result we will look at can be found in Figure 5.8. Here, the GCFA and GCFA-ISP methods have been used to generate growing field with respect to

a *small* working platform. The GCFA method generates a growing field that contain 14 layers and shadow 0 voxels. The growing field of the GCFA-ISP method also contain 14 layers and shadow 0 voxels. In fact, the output of the GCFA-ISP method is in this case equal to the output from the GCFA method. This is to be expected because since the GCFA method never cause any voxels to be shadowed, the ISP improvement scheme never gets called in the method.

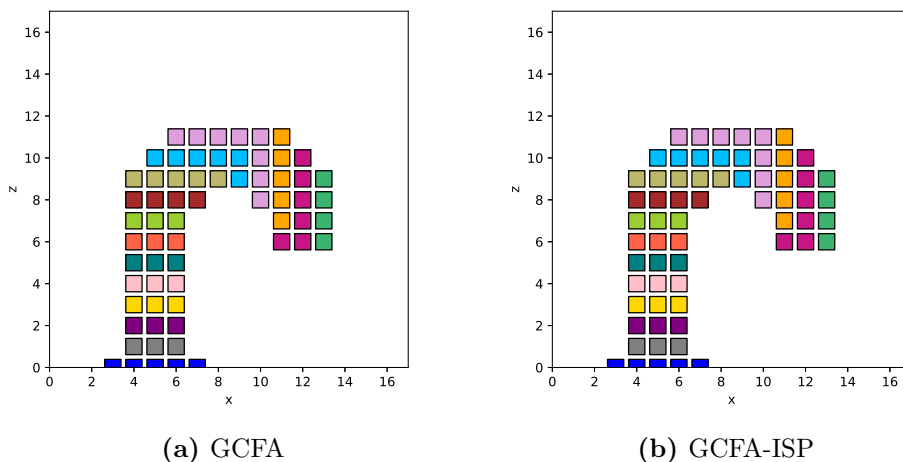
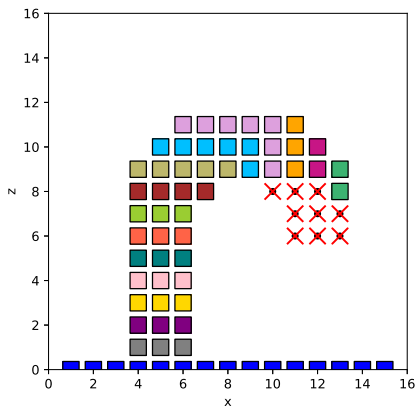


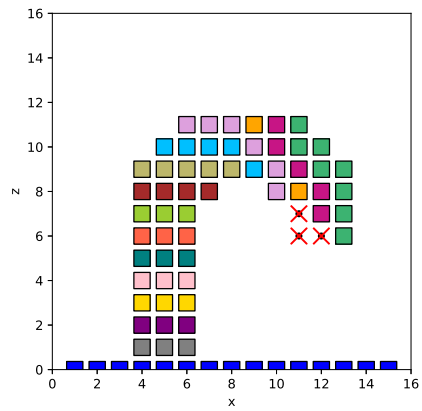
Figure 5.8: Object 1 - growing field \mathcal{G} with respect to small working platform.

The next result we will look at can be found in Figure 5.9, and is generated with respect to a *medium* working platform. Here, the GCFA method generate a growing field that contains 14 layers and shadows 9 voxels. The growing field generated by the GCFA-ISP method contain 14 layers but generate only 3 shadowed voxels.

The third and final result generated with this object can be found in Figure 5.10. Here a *large* platform have been used. We see that the amount of shadowed voxels with both methods have increased significantly compared to the results in both Figure 5.9 and Figure 5.8. The growing field generated by the GCFA method with respect to the large working platform contains 12 layers and shadows 16 voxels. Again, the GCFA-ISP method improves the result and contains 15 layers and shadows 10 voxels.

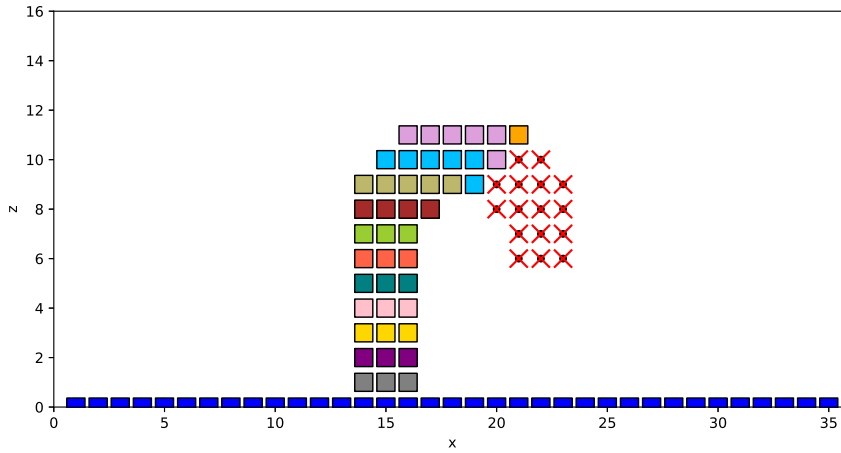


(a) GCFA

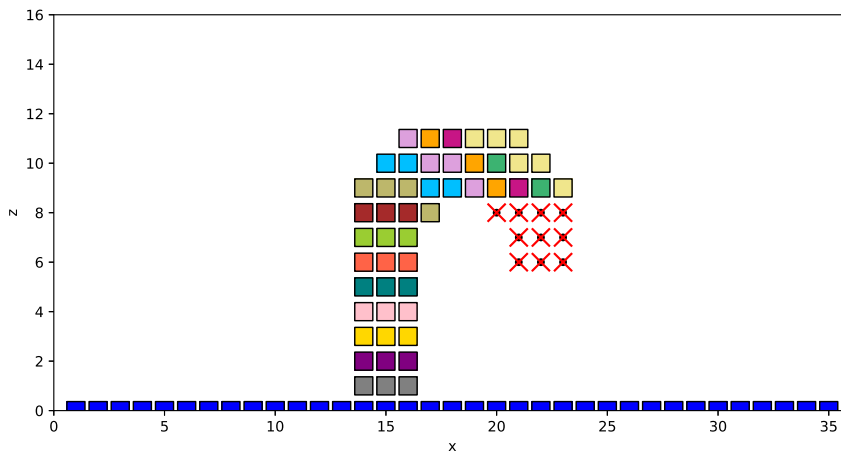


(b) GCFA-ISP

Figure 5.9: Object 1 - growing field \mathcal{G} with respect to medium working platform.



(a) GCFA



(b) GCFA-ISP

Figure 5.10: Object 1 - growing field \mathcal{G} with respect to large working platform.

Object 2

Figure 5.11 show the second test object.

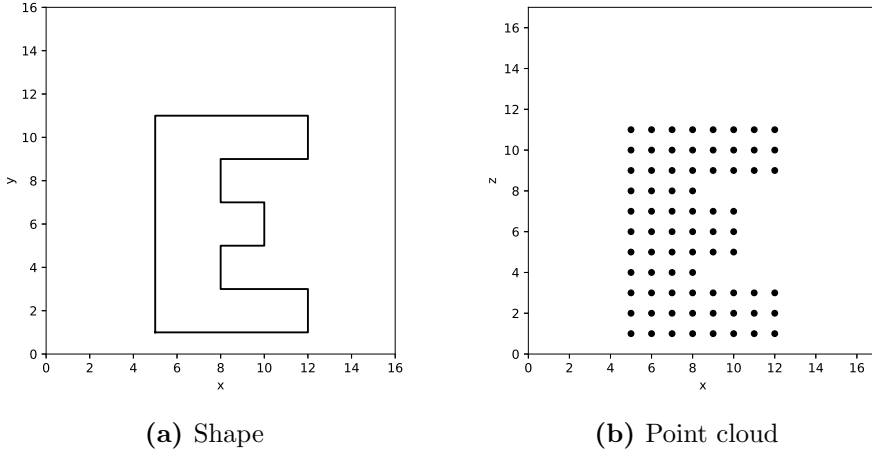
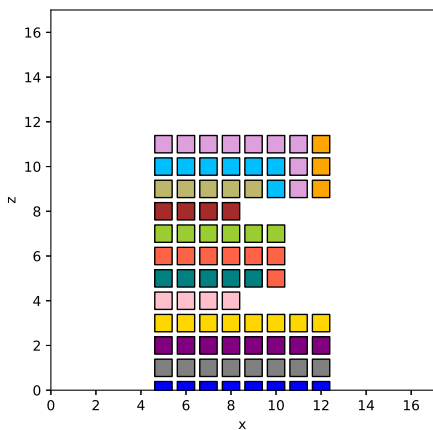


Figure 5.11: Test object 2 represented in two ways.

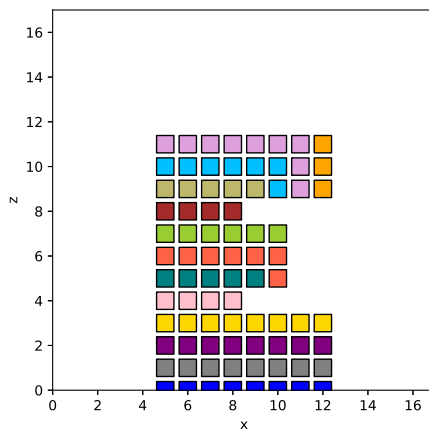
The first results we see, Figure 5.12, is the output from running the programs with object 2 situated on a small working platform as input. Here, the growing field generated with the GCFA method shadows contain 12 layers and shadow 0 voxels. Since the shadowed voxel count is zero, the growing field generated by the GCFA-ISP method is equal to the one generated with the GCFA method.

The next result we see is in Figure 5.13, where a medium platform is used. Both growing field generated here is equal to the growing field generated with respect to the small platform.

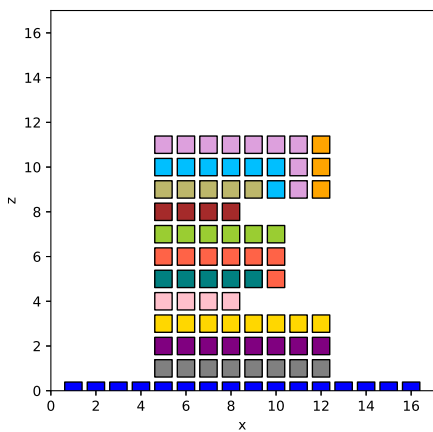
The third and final result of this section can be found in Figure 5.14. Here a large platform have been used. For the first time, voxels of this object have been shadowed. The GCFA method generate 12 layers and shadow 3 voxels, while the GCFA-ISP method generate growing field with 13 layers and shadow 0 voxels.



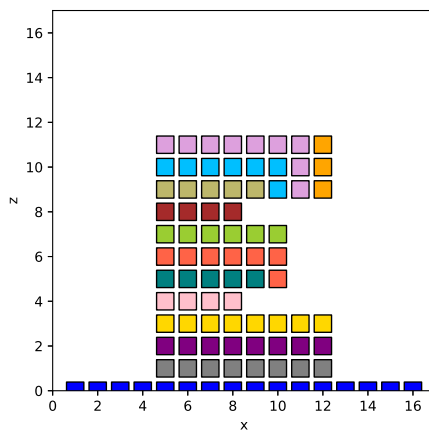
(a) GCFA



(b) GCFA-ISP

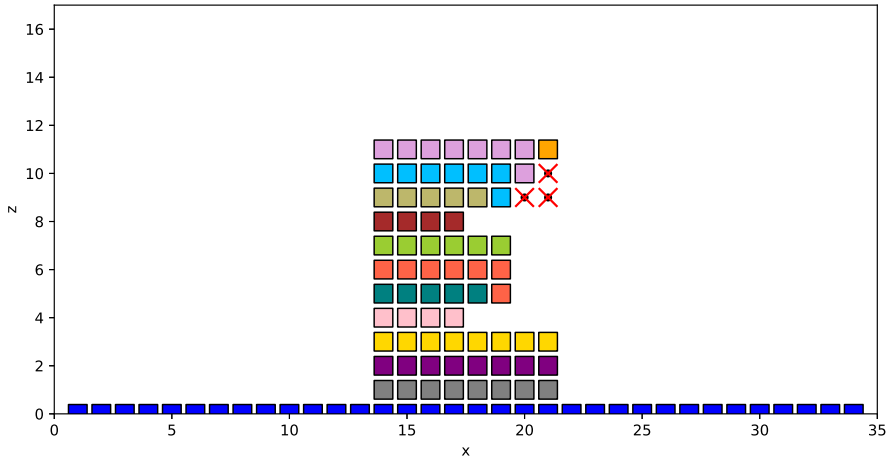
Figure 5.12: Object 2 - growing field \mathcal{G} with respect to small working platform.

(a) GCFA

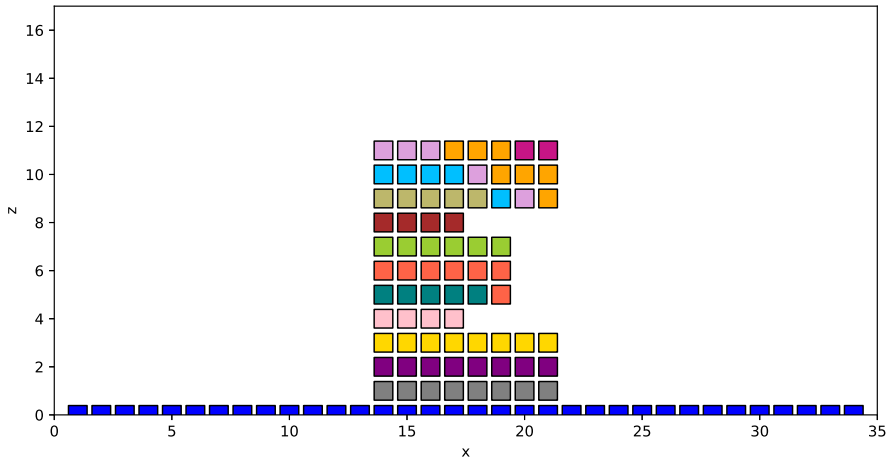


(b) GCFA-ISP

Figure 5.13: Object 2 - growing field \mathcal{G} with respect to medium working platform.



(a) GCEFA



(b) GCEFA-ISP

Figure 5.14: Object 2 - growing field \mathcal{G} with respect to large working platform.

Object 3

Figure 5.15 show the third test object.

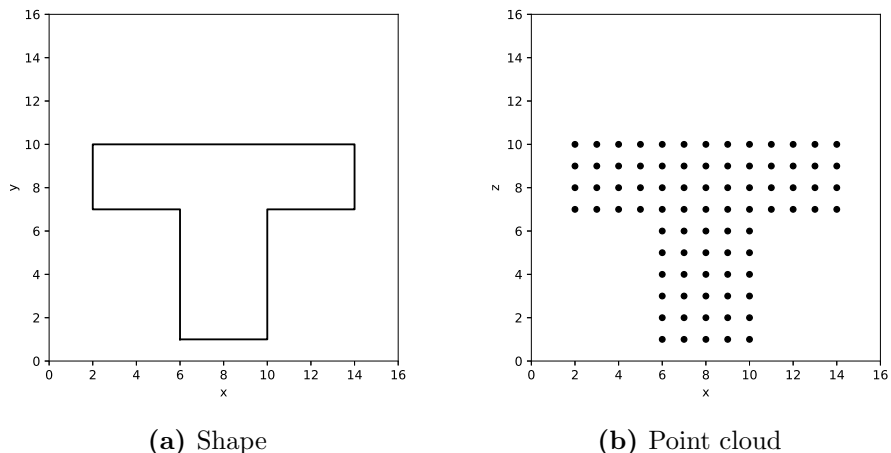
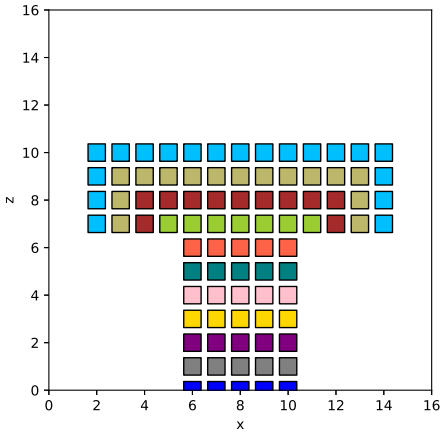


Figure 5.15: Test object 3 represented in two ways.

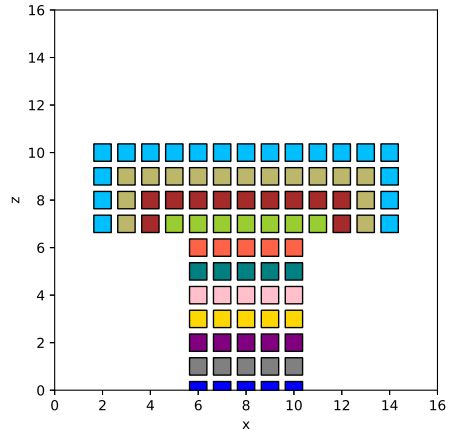
The first results we see, Figure 5.16, is the output from running the programs on object 3 with a small working platform. No voxels are shadowed in this case and the growing field contain 10 manufacturing layers.

The next results we see in Figure 5.17 where a medium platform is used. We obtain the same result in this case, no voxels shadowed and 10 layers.

The third and final result of this section can be found in Figure 5.18, where a large platform have been used. We see that using the GCFA method, 3 voxels on each side of the object i shadowed, i.e. a total of 6 shadowed voxels. The growing field have also in this case got 10 layers. Using the GCFA-ISP the result is improved and we get 0 shadowed voxels and 11 layers.

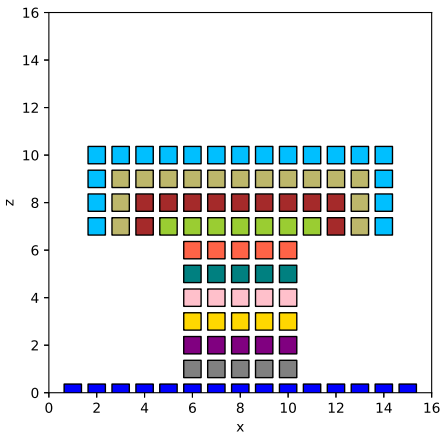


(a) GCFA

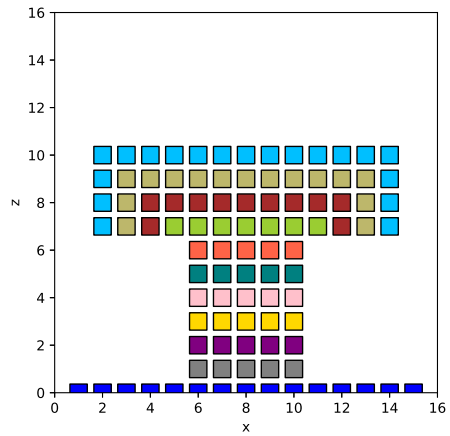


(b) GCFA-ISP

Figure 5.16: Object 3 - growing field \mathcal{G} with respect to small working platform.

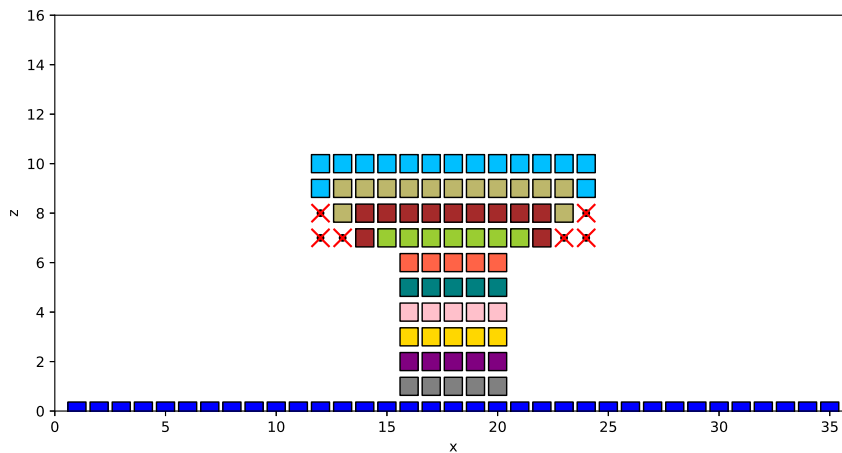


(a) GCFA

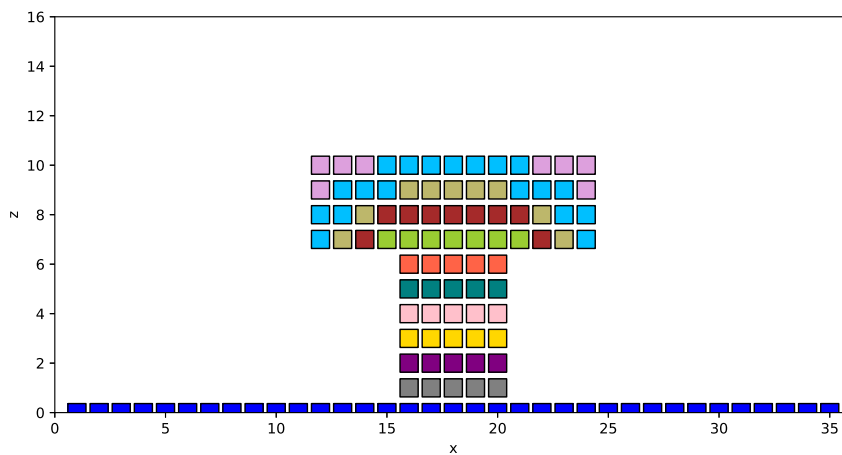


(b) GCFA-ISP

Figure 5.17: Object 3 - growing field \mathcal{G} with respect to medium working platform.



(a) Greedy



(b) Greedy with shadow prevention

Figure 5.18: Object 3 - growing field \mathcal{G} with respect to large working platform.

Object 4

Figure 5.19 show the fourth test object. Simulations have been carried out in order to generate test outputs for both GCFA and GCFA-ISP methods.

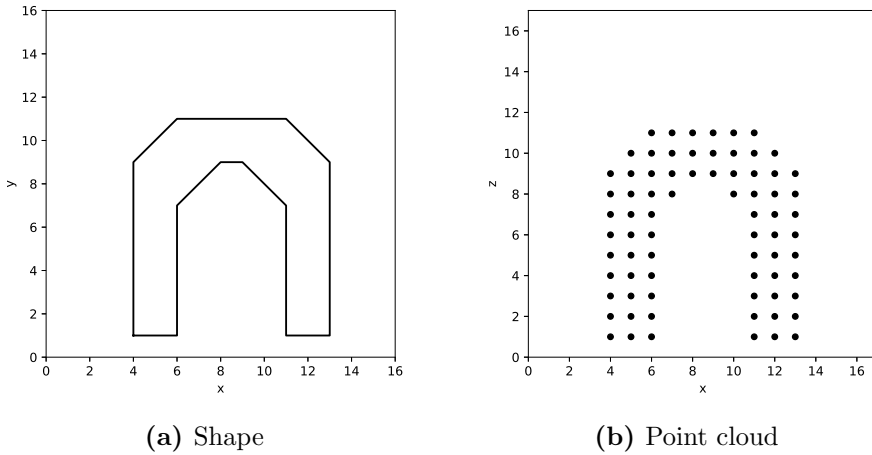
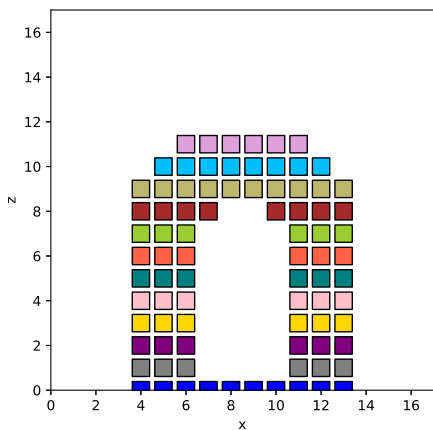


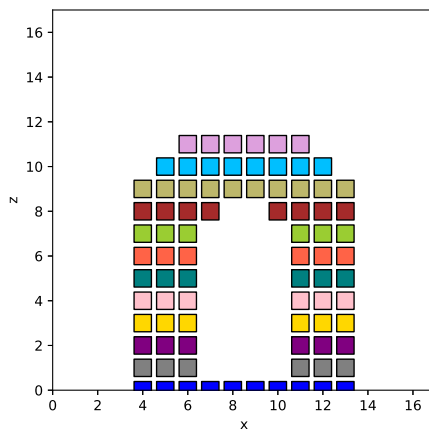
Figure 5.19: Test object 4 represented in two ways.

The first results we see, Figure 5.20, is the output from running the programs with object 4 situated on a small working platform as input. The next results we see is Figure 5.21 where a medium platform is used as input together with the object. The third and final result of this section can be found in Figure 5.22. Here a large platform have been used.

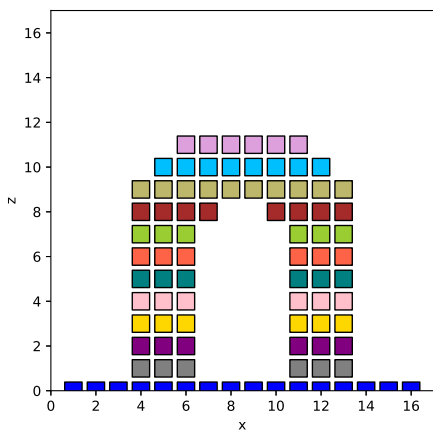
All resulting growing fields of this test object give perfect result, i.e. no shadowed voxels, and all layers are the same.



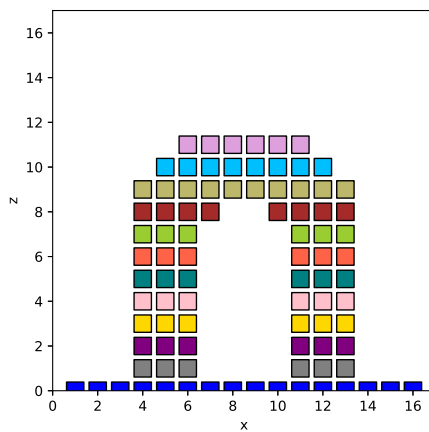
(a) GCFA



(b) GCFA-ISP

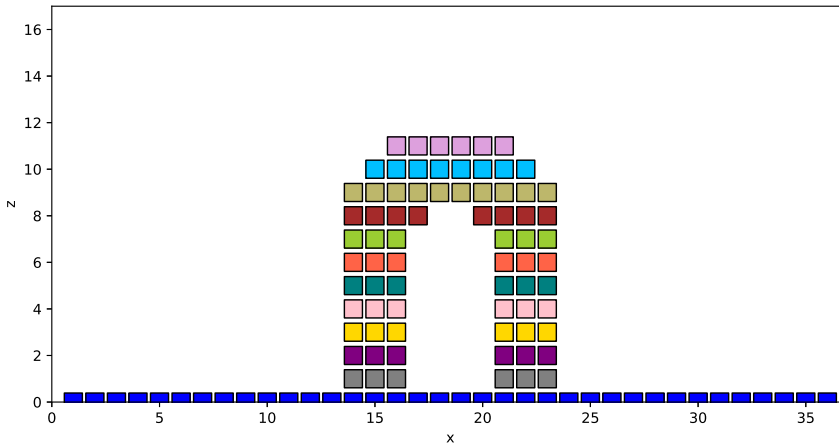
Figure 5.20: Object 4 - growing field \mathcal{G} with respect to small working platform.

(a) GCFA

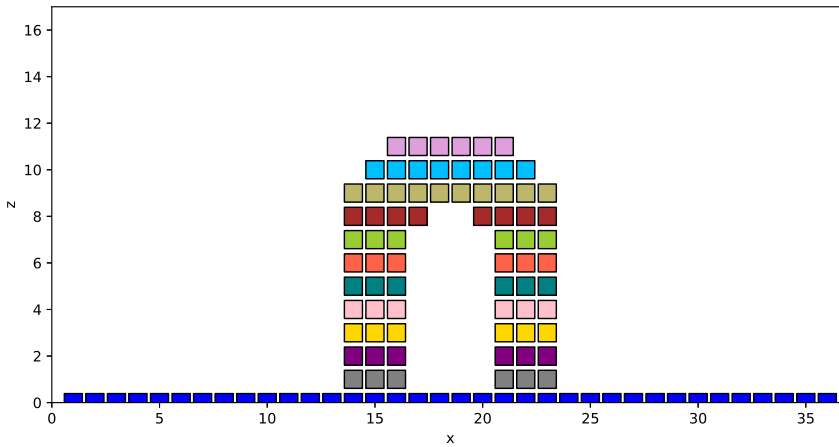


(b) GCFA-ISP

Figure 5.21: Object 4 - growing field \mathcal{G} with respect to medium working platform.



(a) GCFA



(b) GCFA-ISP

Figure 5.22: Object 4 - growing field \mathcal{G} with respect to large working platform.

Object 5

Figure 5.23 show the fifth test object. Simulations have been carried out in order to generate test outputs for both GCFA and GCFA-ISP methods.

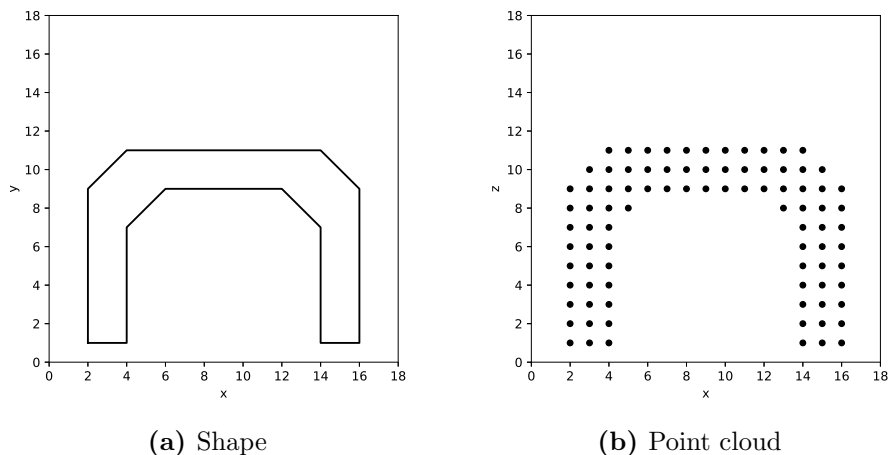


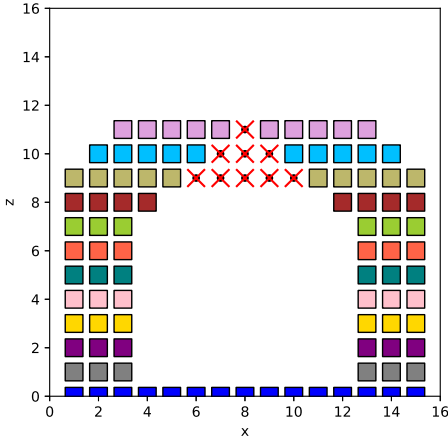
Figure 5.23: Test object 5 represented in two ways.

The first results we see, Figure 5.24, is the output from running the programs with object 5 on a small working platform. The GCFA program generates 11 layers and shadows 9 voxels. The GCFA-ISP program fails to improve the greedy result and generates a growing field containing 21 layers and that leaves 15 voxels shadowed.

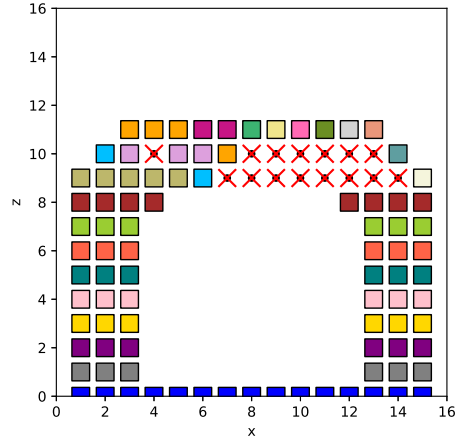
The next results we see is Figure 5.25 where a medium platform is used as input together with the object. The results on this platform is identical to the results with respect to the small platform.

The third and final result of this section can be found in Figure 5.26 where a large platform have been used when generating the growing field. Same as the small and medium platform, the GCFA program generates a growing field of 11 layers and 9 shadowed voxels. Also this time the GCFA-ISP method fails to improve the result and ends up with a growing field of 19 layers that shadows 17 voxels.

It is clear to see that this method is not suited for this type of objects, the reason for this will discussed in Chapter 6.

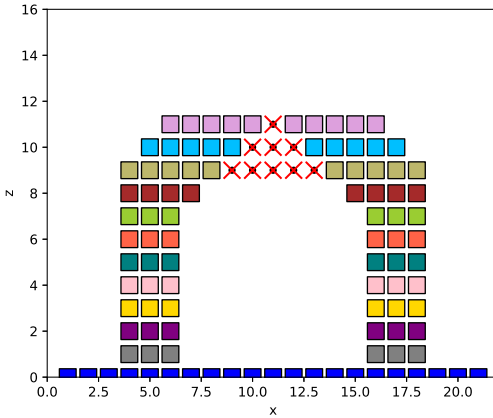


(a) GCFA

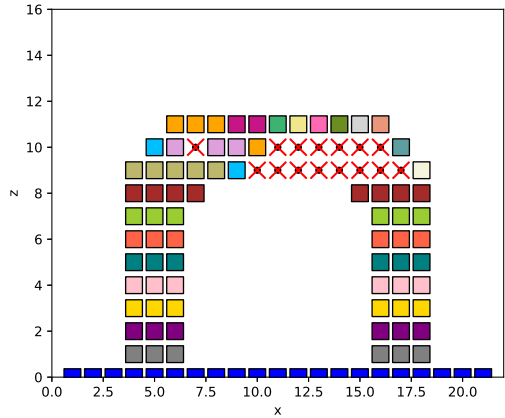


(b) GCFA-ISP

Figure 5.24: Object 5 - growing field \mathcal{G} with respect to small working platform.

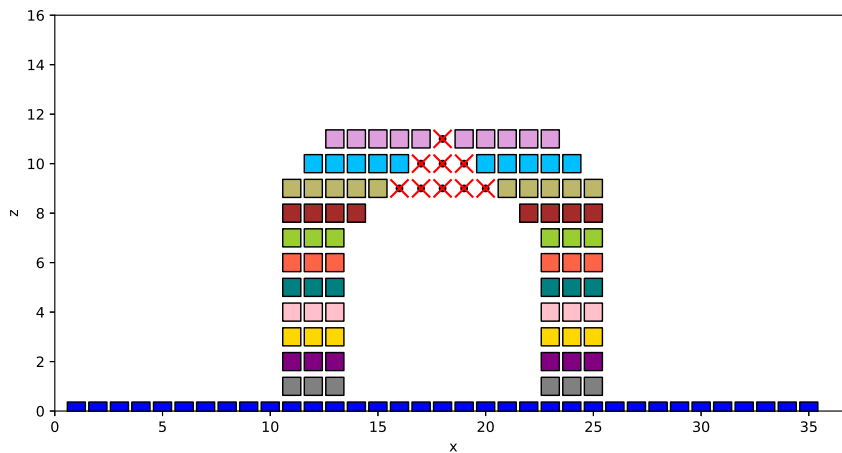


(a) GCFA

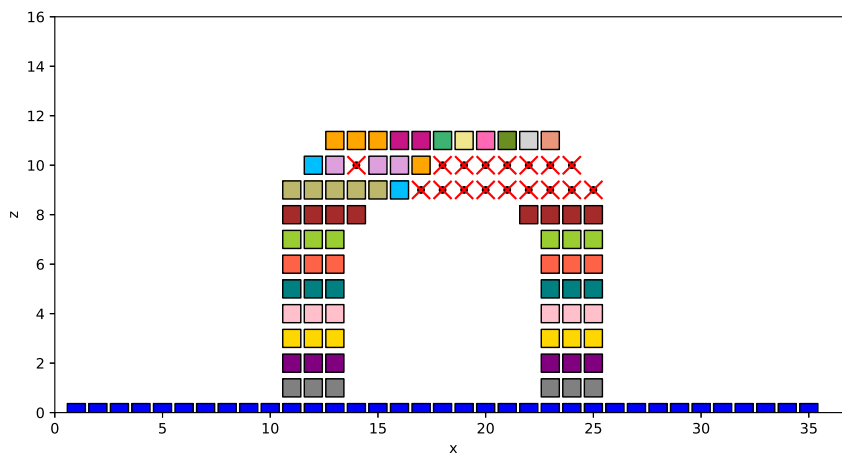


(b) GCFA-ISP

Figure 5.25: Object 5 - growing field \mathcal{G} with respect to medium working platform.



(a) GCFA



(b) GCFA-ISP

Figure 5.26: Object 5 - growing field \mathcal{G} with respect to large working platform.

Object 6

Figure 5.27 show the sixth test object. This object is the first 3D object for the programs to be tested on.

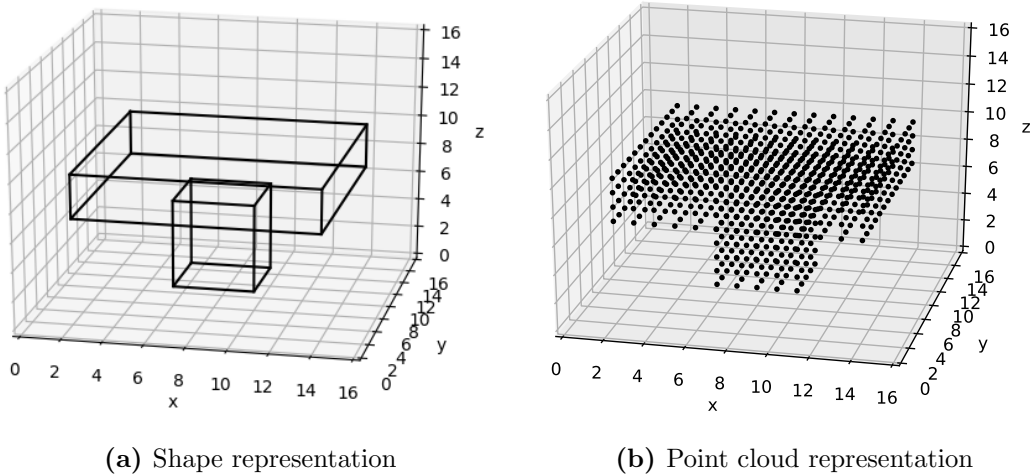


Figure 5.27: Test object 6 represented in two ways.

We start with running the GCFA program on the test object together with a medium working platform. We will soon see that the visualization of the result cannot be made as compact as with the previous 2D object, so we limit the section to visualize only the results that differ from each other and will only name the ones that are equal.

GCFA - Small working platform

Since the result of GCFA with medium working platform below gives a result with 0 shadowed voxels, the small platform will give the same result.

GCFA-ISP - Small working platform

Since the result of GCFA with small platform gives a result with 0 shadowed voxels, the result from running GCFA-ISP will give the same result as with GCFA.

GCFA - Medium working platform

Figure 5.28 show the medium working platform together with the growing field generated by running the GCFA method on object 6 with this platform. Both results above generated by using a small working platform is *identical* to the result in this figure and is therefore not visualized when mentioned above. The generated growing field is symmetric around the axes we see in Figure 5.29.

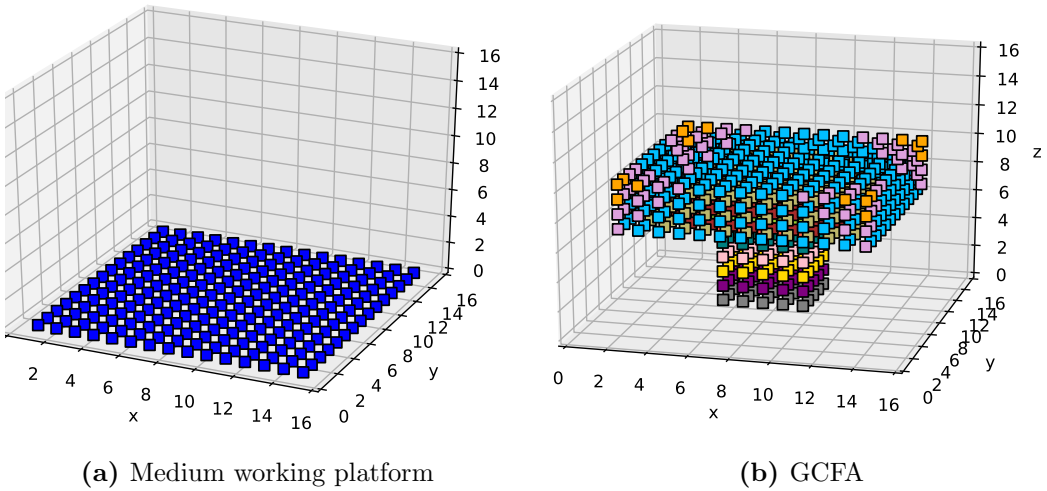


Figure 5.28: The result from running the GCFA program on object 6 together with a medium working platform.

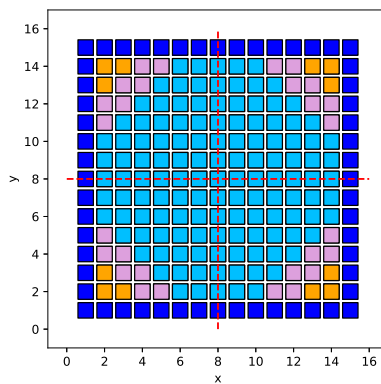


Figure 5.29: Axes of symmetry of growing fields. Object seen from above

In order to show the layers of the solution in the best way possible and to be able to evaluate the solution, we choose to visualize selected cross sections of the growing field, see Figure 5.30 and Figure 5.31.

Figure 5.30 show cross sections at fixed y -values. Since the solution is symmetric around the axes in Figure ??, so only cross sections for "half" the object is shown, i.e. $y = 2$ to $y = 7$. The other "half" of the objects will be identical. An interesting to note is that the plots for values $y = 6$ and $y = 7$ is identical to the growing field of object 3 for small and medium working platform, see Figure 5.16 and Figure 5.17. Figure 5.31 show cross sections at fixed z -values.

This growing field shadows 0 voxels and contains 12 manufacturing layers.

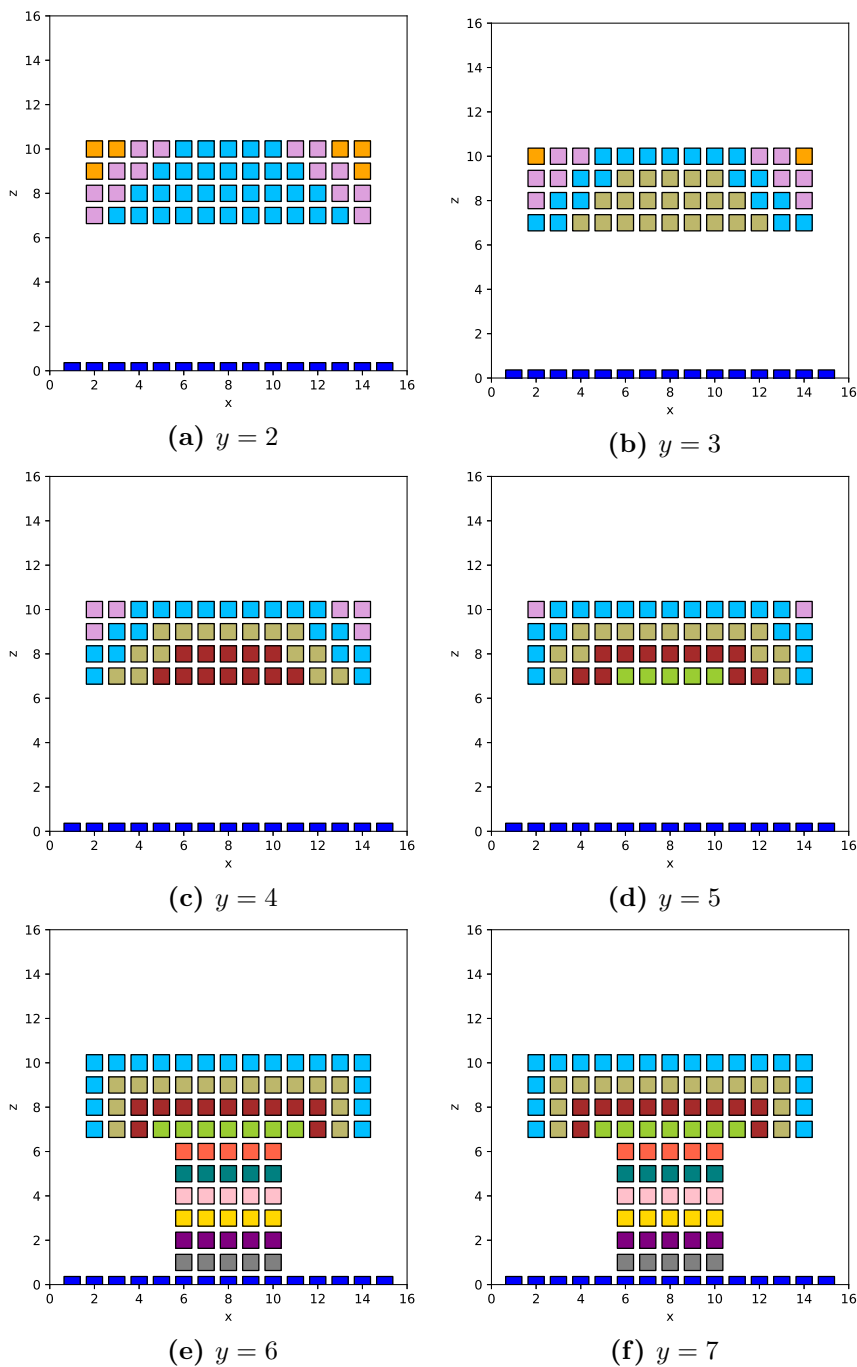


Figure 5.30: Results of GCFA w.r.t. working platform of medium size. Each sub-figure show a cross section of the growing field at constant y -values.

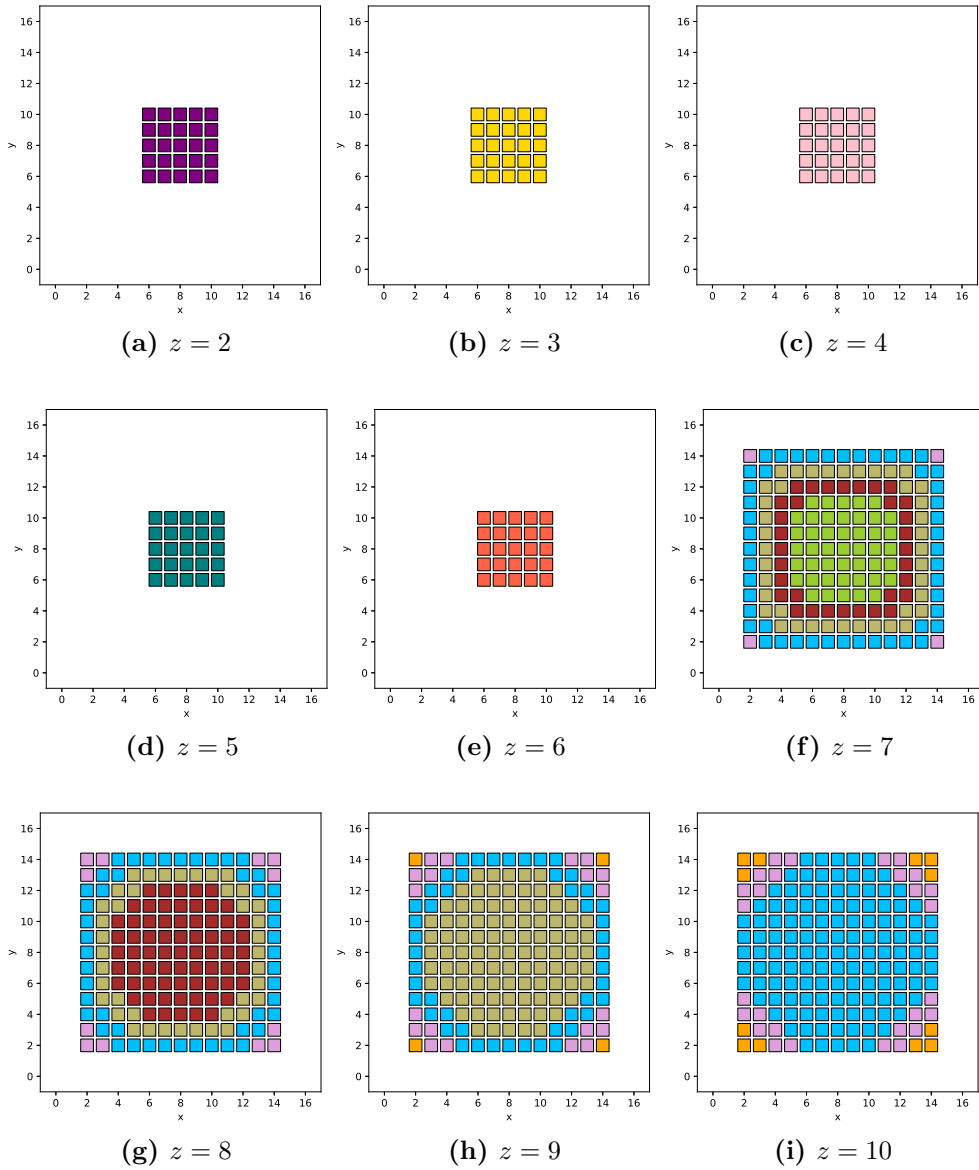


Figure 5.31: Results of GCFA w.r.t. working platform of medium size. Each sub-figure show a cross section of the growing field at constant z -values.

GCFA - Large working platform

The next result we will have a look at is found in Figure 5.32 where a large platform is used as input together with the object and the GCFA program have been run in order to generate the growing field.

Figure 5.32 show the large working platform together with the resulting growing field with respect to the current platform. This solution is also symmetric and the axes of symmetry can be found in Figure 5.33.

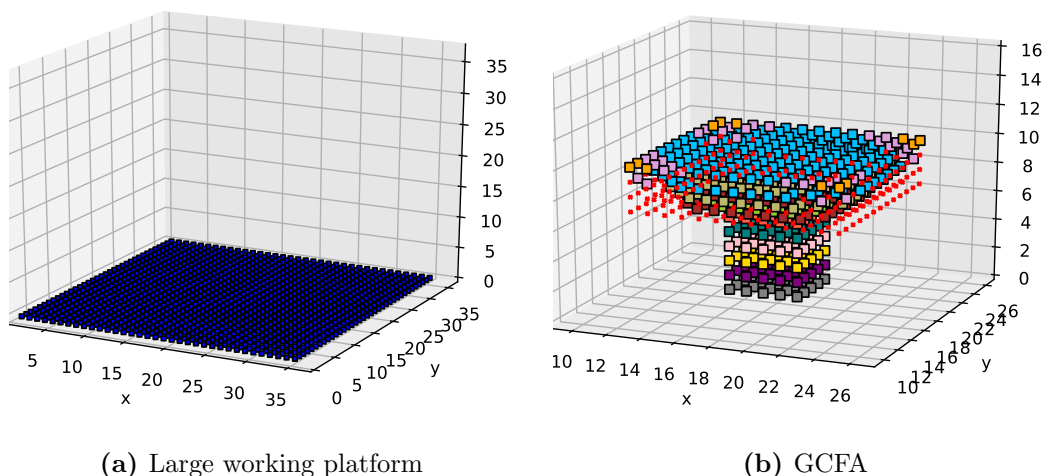


Figure 5.32: The result from running the GCFA program on object 6 together with a large working platform.

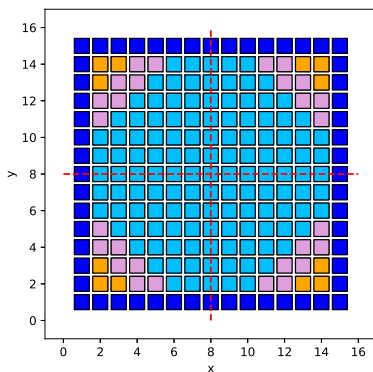
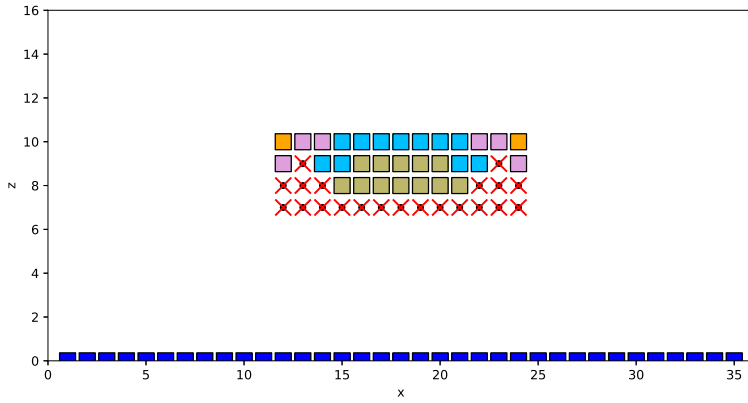
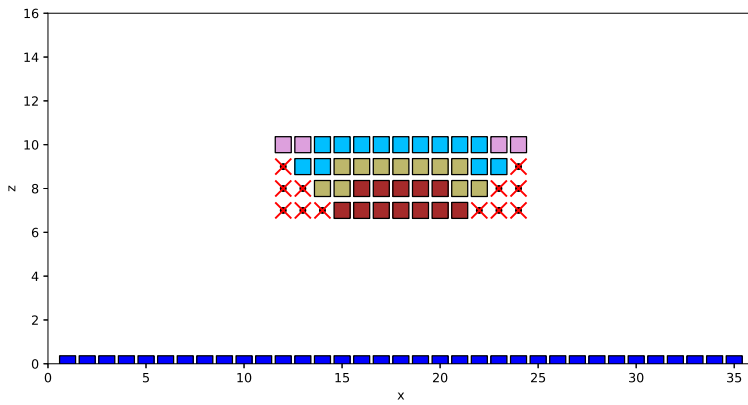


Figure 5.33: Axes of symmetry of growing fields. Object seen from above

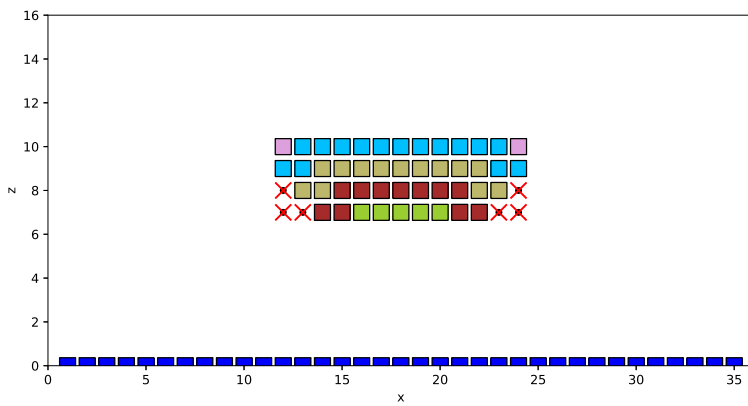
Same as for the medium platform results, the growing field is visualized as cross sections at fixed y - and z -values, see Figure 5.34 and Figure 5.35.



(a) $y = 13$



(b) $y = 14$



(c) $y = 15$

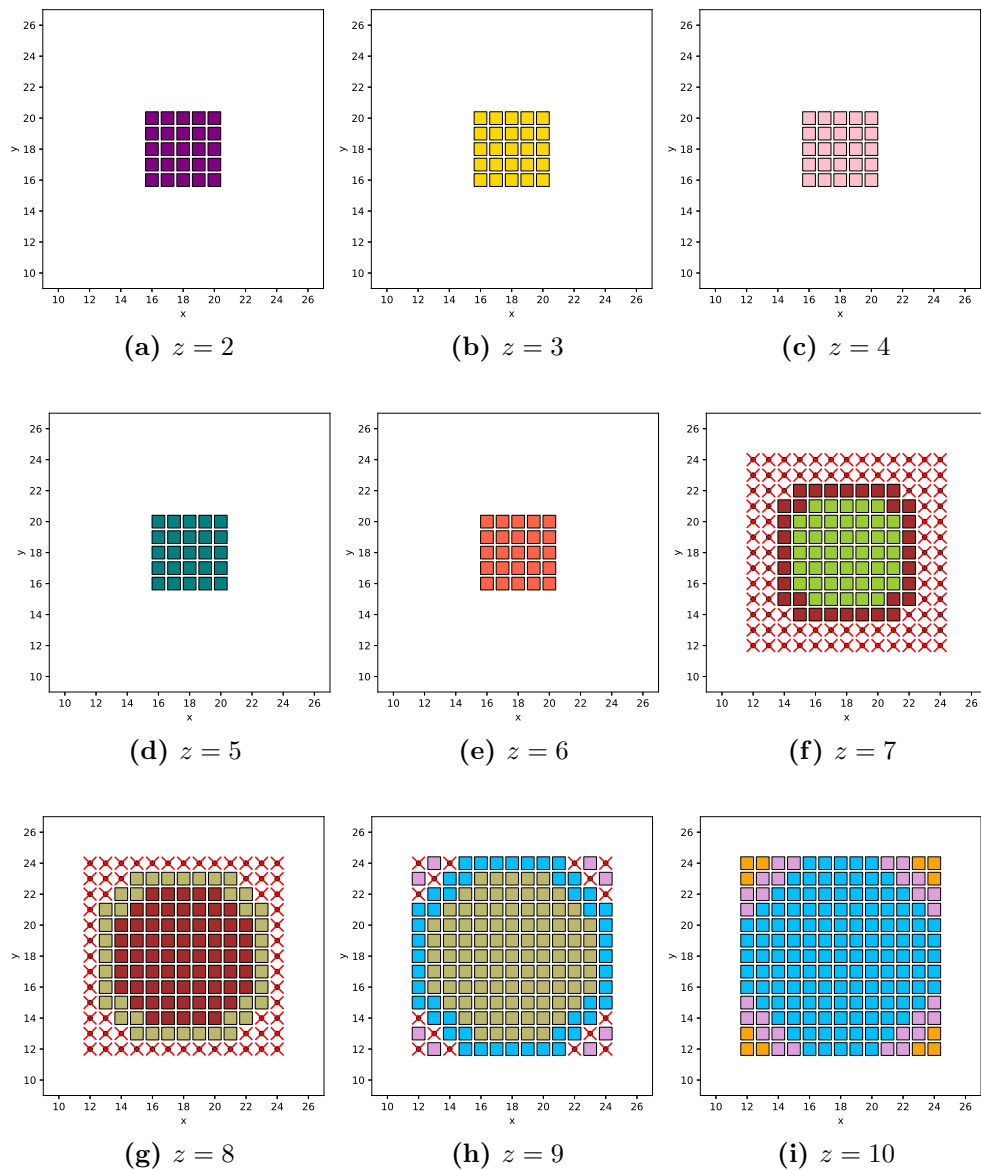
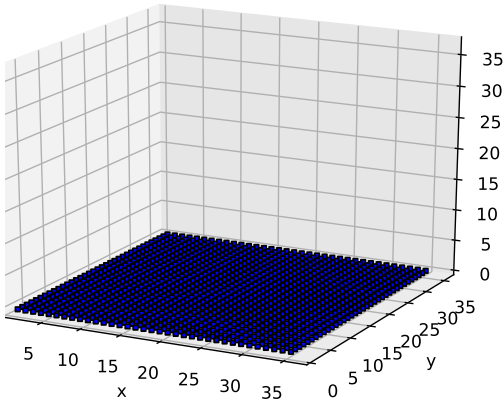


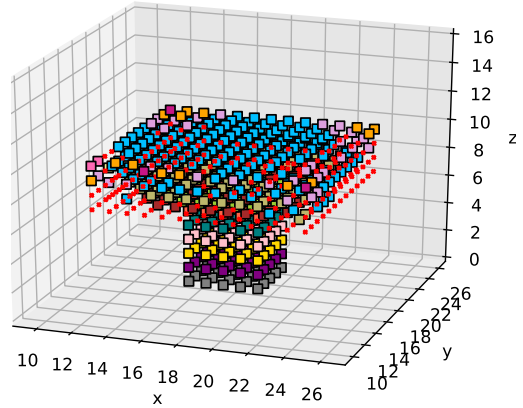
Figure 5.35: Results of GCFA w.r.t. working platform of large size. Each sub-figure show a cross section of the growing field at constant z -values.

GCFA-ISP - Large working platform

Figure 5.36b show the large working platform together with the resulting growing field with respect to the current platform.



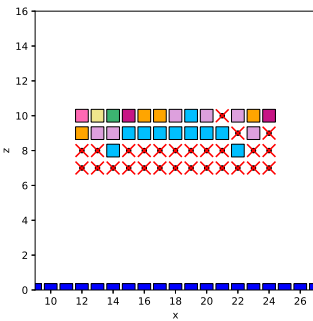
(a) Large working platform



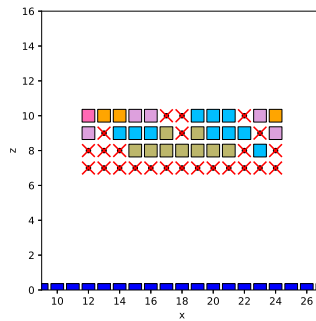
(b) GCFA-ISP

Figure 5.36: The result from running the GCFA-ISP program on object 6 together with a large working platform.

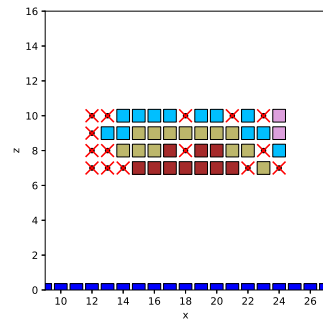
Same as for the medium platform results, the growing field is visualized as cross sections at fixed y - and z -values, see Figure 5.37 and Figure 5.38.



(a) $y = 12$



(b) $y = 13$



(c) $y = 14$

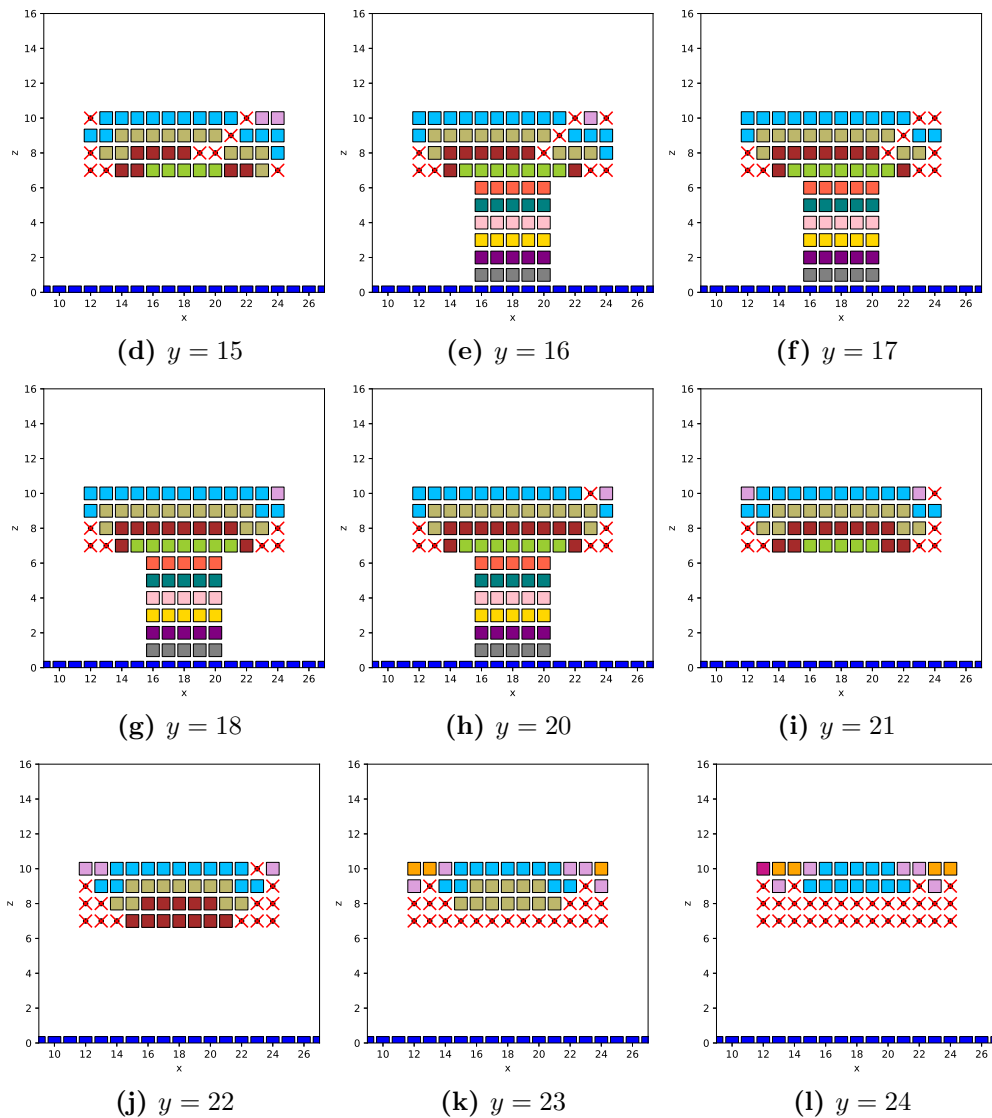


Figure 5.37: Results of GCFA-ISP w.r.t working platform of large size.

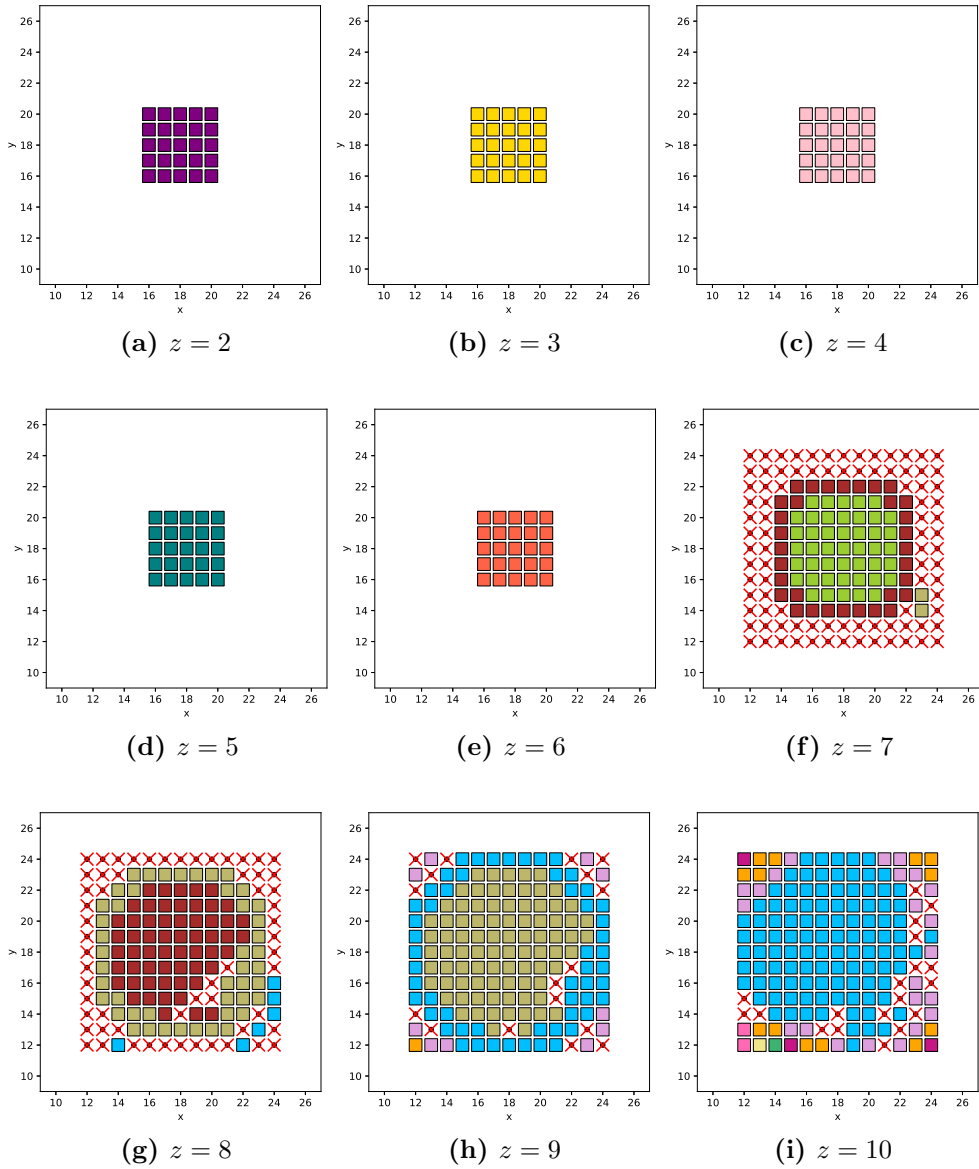


Figure 5.38: Results of GCFA w.r.t. working platform of large size. Each sub-figure show a cross section of the growing field at constant z -values.

5.2.3 Summary of results

In this section, the results presented in Section 5.2.2 is summarized using two tables. Table 5.1 shows the number of missed voxels generated by each method when run on the objects with respect to the different platform sizes, while table 5.2 show the number of generated layers. The results will be discussed in Chapter 6.

Table 5.1: Algorithm performance in terms of missed voxels.

Number of missed voxels						
Platform	S		M		L	
Method	GCFA	GCFA-ISP	GCFA	GCFA-ISP	GCFA	GCFA-ISP
<i>Object</i>						
1	0	0	9	3	16	10
2	0	0	0	0	3	0
3	0	0	0	0	6	0
4	0	0	0	0	0	0
5	9	15	9	15	9	17
6	0	0	0	0	168	185

Table 5.2: Algorithm performance in terms of manufacturing layers.

Number of manufacturing layers						
Platform	S		M		L	
Method	GCFA	GCFA-ISP	GCFA	GCFA-ISP	GCFA	GCFA-ISP
<i>Object</i>						
1	14	14	14	14	12	15
2	12	12	12	12	12	13
3	10	10	10	10	10	11
4	11	11	11	11	11	11
5	11	21	11	21	11	19
6	12	12	12	12	12	16

Chapter 6

Discussion

In this chapter the algorithms, the simulation results and the hardware will be thoroughly discussed. Section 6.1 summarizes the changes made to the algorithms from the paper and explains why they were made. Section 6.2 discusses the results presented in Chapter 5. Section 6.3 compares the results obtained in this thesis to traditional AM decomposition methods. Section 6.4 discusses the hardware used in the paper, common robotic AM configurations and proposes a new hardware based on the two above.

6.1 Changes made to the original algorithms

The original algorithms from [3][54] for decomposing objects into curved manufacturing layers is found in Appendix B. As mentioned in Chapter 4, some changes had to be made to the original algorithms in order to obtain realizable and satisfying results. The specific changes will be reviewed and discussed in this section.

6.1.1 Defining AM-stable neighbors (ASN)

In [3], the set of AM stable neighbors is defined as in Definition 6 below. Figure 6.1a show an illustration of this set in 2D. For comparison, the illustration showing the set of ASNs proposed in Chapter 4 of this thesis, Definition 3, can be found in Figure 6.1b.

Definition 6. Two voxels, $v_{i,j,k}$ and $v_{r,s,t}$, are defined as *AM-stable-neighbours* (ASN) if $\|(i, j, k) - (r, s, t)\|_1 \in \{1, 2\}$

In the algorithms, the set of ASNs represent voxels eligible for being accumulated into the next layer, \mathcal{L}_{next} , because they can be manufactured in a support-free manner. The difference between Definition 6 and Definition 3 is demonstrated in Figure 6.1. We see that some points in Figure 6.1a is not present in Figure 6.1b. The change made to Algorithm 3 is that these four points (in 3D, six points) have been taken out of the set because they enable for the adding of voxels that violate the constraint for support-free for manufacturing, Constraint 1 in Chapter 4.

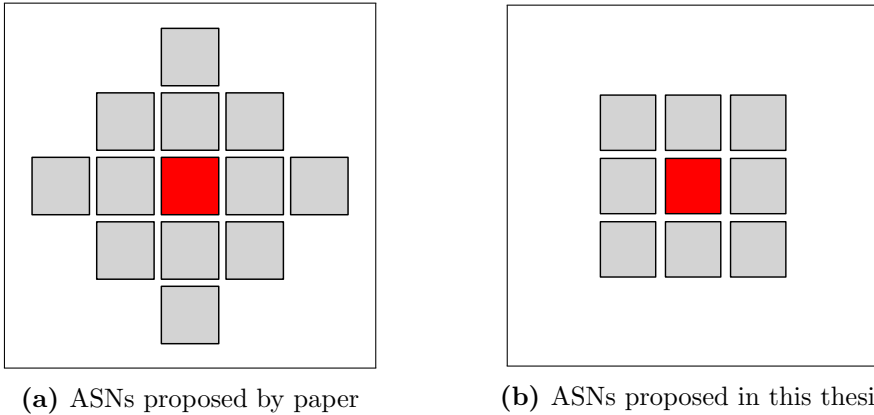


Figure 6.1: Results using two different choices of norm-values when defining the set of ASNs.

6.1.2 ISP voxel accumulation threshold

The ISP improvement scheme incrementally calculates new convex fronts in order to find specific voxels that cause new voxels to be shadowed by looking one step ahead.

In the code-snippet below, Code 6.1, from the implementation of `incremental_shadow_prevention(...)`, `test_shadowed` is a list containing voxels shadowed by `test_front` (if any). `potential_shadowed` is the list of voxels shadowed by the convex front calculated by the greedy scheme that called the ISP-method and

that was used to generate `next_layer` which ISP aim to improve.

Throughout the run of `incremental_shadow_prevention(..)`, `potential_shadowed` stays the same, while the elements of `test_shadowed` change. In order to determine if a voxel should be added to the improved (reduced) next layer, $\tilde{\mathcal{L}}_{next}$, we can compare the length of `test_shadowed`, `len(test_shadowed)`, to a numeric value. This numeric value can be seen as a threshold of whether to add a voxel or not. Line 11 of Algorithm 4 in the Appendix states:

11: Add v into $\tilde{\mathcal{L}}$ if $\mathcal{S}_t = \mathcal{S}_c$

In Algorithm 2, which is the altered version of Algorithm 4, we instead write:

22: **If** \mathcal{S}_t contains less voxels than \mathcal{S}_p **then**

23: Add v to $\tilde{\mathcal{L}}_{next}$

The differences between these two instructions is that the line from Algorithm 4 is weaker than the instruction from Algorithm 2. For Algorithm 2 to approve of a voxel being added to $\tilde{\mathcal{L}}_{next}$, the voxel must cause the shadowing of *less* voxels than what is currently being shadowed by \mathcal{L}_{next} proposed by the greedy scheme, which is `len(potential_shadowed)`. The line from Algorithm 4 however, allows for the accumulation of voxels that cause the shadowing of the same amount of voxels as \mathcal{L}_{next} .

So line 11 in Algorithm 4 have been replaced by line 22 and 23 in Algorithm 2. Code 6.1 show the implementation of this part of the program.

```
test_shadowed = find_shadowed(model, test_front, test_processed)

if len(test_shadowed) < len(potential_shadowed):
    reduced_next_layer.append(voxel)
```

Code 6.1: Checking whether to keep a voxel in the improvement scheme.

6.2 Simulation results

In this section, the results presented in Section 5.2 will be discussed. For reference, Table 6.1 show an overview of where the simulation figures of each object, platform and method can be found in the thesis.

Table 6.1: An overview of where to find which simulation result in the thesis.

Simulation result figures						
Platform	S		M		L	
Method	GCFA	GCFA-ISP	GCFA	GCFA-ISP	GCFA	GCFA-ISP
<i>Object</i>						
1	5.8a	5.8b	5.9a	5.9b	5.10a	5.10b
2	5.12a	5.12b	5.13a	5.13b	5.14a	5.14b
3	5.16a	5.16b	5.17a	5.17b	5.18a	5.18b
4	5.20a	5.20b	5.21a	5.21b	5.22a	5.22b
5	5.24a	5.24b	5.25a	5.25b	5.26a	5.26b
6	–	–	5.28b	–	5.32b	5.36b
			5.30		5.34	5.37
			5.31		5.35	5.38

6.2.1 Note: 2D vs. 3D objects

The methods from Chapter 4 are developed for dividing 3D objects into manufacturing layers, so we start the discussion by commenting on why the results from this thesis are for the most part limited to 2D object, with only one exception, i.e. object 6. The basis for using mostly 2D objects to test the algorithms is that a 3D body is, in reality, a set of 2D objects stacked either on top of each other or next to each other, a concept demonstrated in Section 5.1.2 and used for visualization in Section 5.2.2. Due to this, the results from the 2D testing are still valid in 3D and is used for demonstrating the concept and to evaluate the algorithms against each other and on different types of objects.

The same implementation is used for both 2D and 3D input objects. So after using 2D objects to develop an understanding of the algorithms and to identify strengths and weaknesses in the programs, large-scale experimentation can be made on 3D objects to verify the results derived from the 2D experiments. In order to efficiently perform tests on 3D objects, an automated method for importing objects from CAD software or similar methods must be developed. More on this topic in the section proposing further work, i.e. Section 7.2.

6.2.2 Looking at the result tables

In order to discuss the results of the performed simulations presented in Chapter 5, we restate the tables summarizing the quantitative measurements of the results

from Section 5.2.3 in Table 6.2 and Table 6.3 in this section. A desired result can be measured as having a low amount of missed voxels and a low amount of manufacturing layers.

We start by discussing the findings of Table 6.2, i.e. the number of missed voxels. Missed voxels are the sum of shadowed voxels and voxels omitted without being shadowed. The shadowing of voxels is caused by the size of the current convex front \mathcal{C}_c . Voxels that never enter the set of ASNs can never be considered as support-free for manufacturing and is omitted due to this.

Looking at Table 6.2, we see that all growing fields are affected by the increase in platform size. For every object, the worst result, i.e. the result with the largest amount of missed voxels, is obtained by using a large platform. For every object except object 5, the resulting growing field with respect to the small platform, have zero missed voxels. Why the growing field generation is influenced by the platform size will be discussed Section 6.2.3.

We also see that, in general, the ISP-algorithm does result in a fewer number of missed voxels. The GCFA-ISP program manages to improve the result for all objects except object 5. Why we obtain poor results for this particular object is also discussed in Section 6.2.3.

Table 6.2: Comparison of algorithm performance in terms of missed voxels

		Missed voxels					
Platform		S		M		L	
Method		GCFA	ISP-GCFA	GCFA	ISP-GCFA	GCFA	ISP-GCFA
<i>Object</i>							
	1	0	0	9	3	16	10
	2	0	0	0	0	3	0
	3	0	0	0	0	6	0
	4	0	0	0	0	0	0
	5	9	15	9	15	9	17
	6	0	0	0	0	168	185

Now we will have a look at what the layer count can tell us about a solution. Due to adhesion between layers and the finishing look of the realized object, it is desirable that the method generates as few manufacturing layers as possible. If two methods generate growing fields that have the same amount of missed voxels,

the growing field with the least amount of manufacturing layers would be the best result.

In Table 6.3, we see that for object 1-3 the layer count only changes when testing with the largest platform. In object 4 the layer count is the same for all methods, while for object 5 the ISP method generates almost double the layers of the GCFA methods for all platforms.

Table 6.3: Comparison of algorithm performance in terms of manufacturing layers

Number of manufacturing layers						
Platform	S		M		L	
Method	GCFA	GCFA-ISP	GCFA	GCFA-ISP	GCFA	GCFA-ISP
<i>Object</i>						
1	14	14	14	14	12	15
2	12	12	12	12	12	13
3	10	10	10	10	10	11
4	11	11	11	11	11	11
5	11	21	11	21	11	19
6	12	12	12	12	12	16

6.2.3 Interpreting the results

From Section 6.2.2 we have experienced that the quality of the results is affected by:

- Object configuration:

It is clear to see that the evaluation of each solution is affected by the properties of each object. E.g. object 5 have poor results for all platform sizes and both methods, while object 4, which is quite similar to object 5 gives good result for every test. Every object is composed of units that form a combination of critical regions. The methods generate good results on some combinations of critical regions and poor results for other combinations. Experimenting can give valuable information on what objects the methods are suited for.

- Platform size:

The platform size affects how the object is divided into manufacturing layers and how many voxels are missed. This happens because of the way the accessible working surface is defined. The definition of the accessible surface was stated back in Section 4.4 as the current convex front, i.e. the convex hull of the previously processed voxels and the working platform. Every point that lies outside of the current convex front is a candidate for being added to the next layer, so since the calculation of the front depends on the platform, the platform will affect which voxels are accumulated. A smaller platform will in most cases result in a smaller front, which in turn leads to a larger amount of next layer candidates among the voxels of the object.

6.2.4 Evaluation of the results

Table 6.4 show a schematic overview of the evaluation of each growing field. Three evaluation groups is used:

- Good: No voxels shadowed and few manufacturing layers
- Ok: Acceptable amount of shadowed voxels and few manufacturing layers
- Poor: Large amount of shadowed voxels and/or many manufacturing layers

Table 6.4: Evaluation of test results

		Result					
Platform		S		M		L	
Method		GCFA	GCFA-ISP	GCFA	GCFA-ISP	GCFA	GCFA-ISP
<i>Object</i>							
1		good	good	poor	ok	poor	poor
2		good	good	good	good	poor	good
3		good	good	good	good	poor	good
4		good	good	good	good	good	good
5		poor	poor	poor	poor	poor	poor
6		good	good	good	good	poor	poor

For the small working platform, all objects except object 5 give "good" results using both the GCFA and the GCFA-ISP method. For the medium working

platform object 2-4 give the best results ("good" for both methods), object 1 gives "ok" result for the GCFA-ISP method while the remaining result is evaluated as "poor". For the large working platform, the only "good" result is for object 2-3 with GCFA-ISP and for object 4 with both methods. The rest are evaluated as "poor" for this platform.

6.2.5 How to improve the results

Since the GCFA method uses a greedy approach the result is expected to not be optimal as mentioned when reviewing the approach in Section 4.4.3, so no further improvements are suggested here other than using the smallest platform which is practical to use when realizing the object.

Improvements to the GCFA-ISP method is, however, possible to obtain better results. The voxel accumulation threshold of the ISP-improvement, discussed in Section 6.1.2 can be tuned, i.e. changed to any numeric value, for the program to give good results for a specific object with respect to the desired platform.

An other alternative for improving the GCFA-ISP result is to investigate different orderings of the list Q containing all elements of the proposed greedy layer \mathcal{L}_{next} subject for improvement by reduction. The reason for the poor results of object 5 and object 6 when using the GCFA-ISP method is the ordering of Q . The authors of [3] which proposed the method, state that different sequences of Q result in different "safe" subsets, and that it is desirable to obtain connected large regions that can be easily covered by tool-paths. The paper algorithm-command for deciding the ordering of Q is:

7: Determine a heuristic sequence Q of voxels in \mathcal{L}_{next} by a flooding algorithm;

The results in this thesis are generated by this command for choosing Q :

8: Copy the values of \mathcal{L}_{next} into Q ;

This choice for Q is made because of the simplicity of the implementation, and as found in Section 6.2.4, this choice gives good results for many of the tests.

6.3 Curved layers vs. planar layers

We learned in Section 2.1.4 about the fixed build direction being a limitation of the commonly used gantry system in AM. As a consequence of the build direction

having to be fixed, objects are usually divided into planar layers perpendicular to the chosen build direction before manufacturing. These steps were previously reviewed in Section 2.1.5.

It is interesting to compare the results obtained by allowing layers to be curved instead of strictly planar. This section will, therefore, compare the *worst* result from Section 5.2 obtained by the new robotic AM approach investigated in this thesis to results obtained by slicing the same test objects into uniform, i.e. equal size, planar layers. Table 6.5 highlights the results from Section 5.2 with the highest amount of missed voxels and are the results we wish to compare to the uniform slicing approach.

Table 6.5: Identification of the worst results obtained using one of the new methods for each object.

Missed voxels						
Platform	S		M		L	
Method	GCFA	ISP-GCFA	GCFA	ISP-GCFA	GCFA	ISP-GCFA
<i>Object</i>						
1	0	0	9	3	16	10
2	0	0	0	0	3	0
3	0	0	0	0	6	0
4	0	0	0	0	0	0
5	9	15	9	15	9	17
6	0	0	0	0	168	185

We leave out the 3D object, object 6 from this comparison because it is sufficient to compare the 2D object to demonstrate what we want to show in this section. The chosen results are visualized together with newly generated results from the test-object being uniformly sliced, see Figure 6.2-6.6. The voxel counts are summarized in Table 6.6 for easy comparison.

It is easy to see that, for all test objects, the worst result using a curved-layer method still produces fewer missed voxels than using uniform slicing.

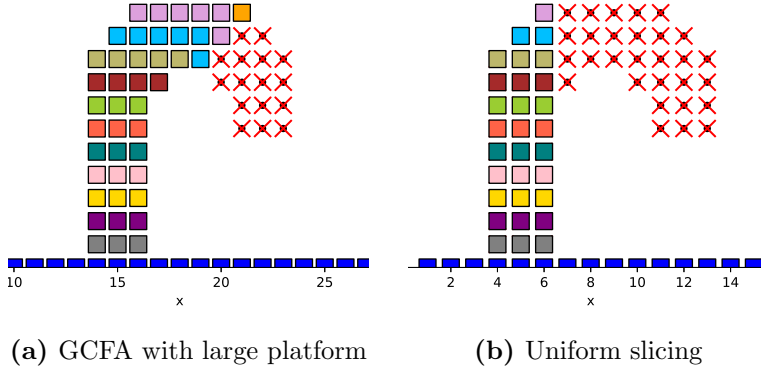


Figure 6.2: Object 1 – The worst result with new method vs. uniform slicing

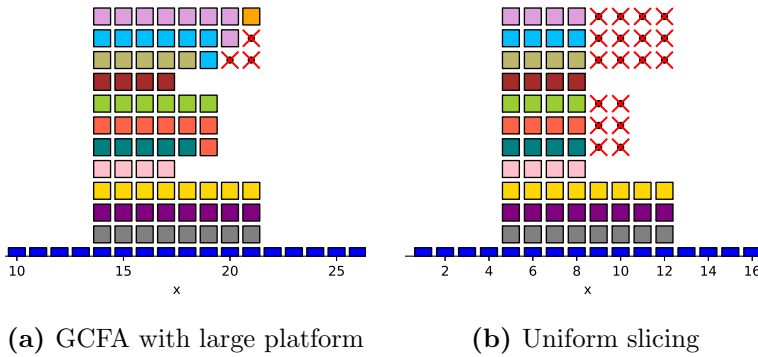


Figure 6.3: Object 2 – The worst result with new method vs. uniform slicing

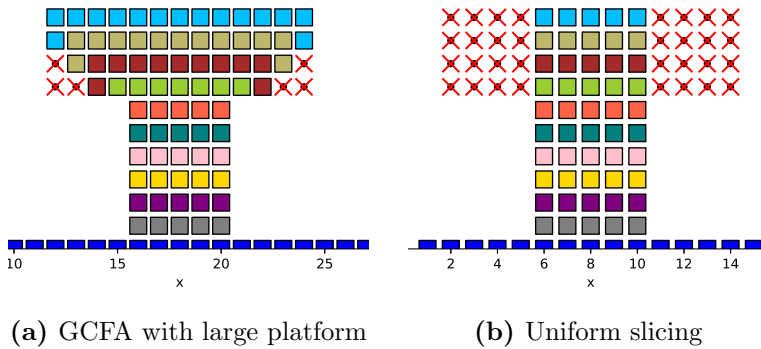


Figure 6.4: Object 3 – The worst result with new method vs. uniform slicing

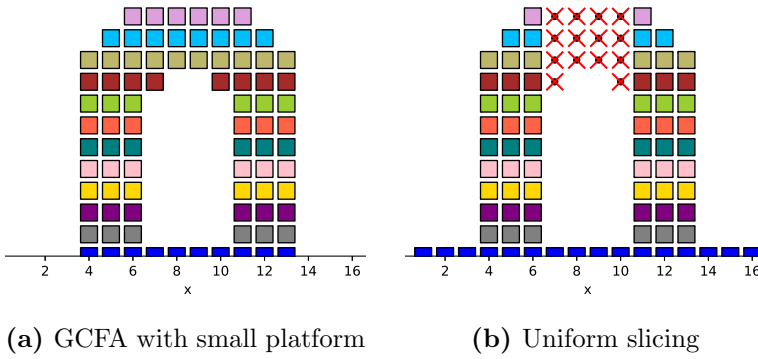


Figure 6.5: Object 4 – The worst result with new method vs. uniform slicing

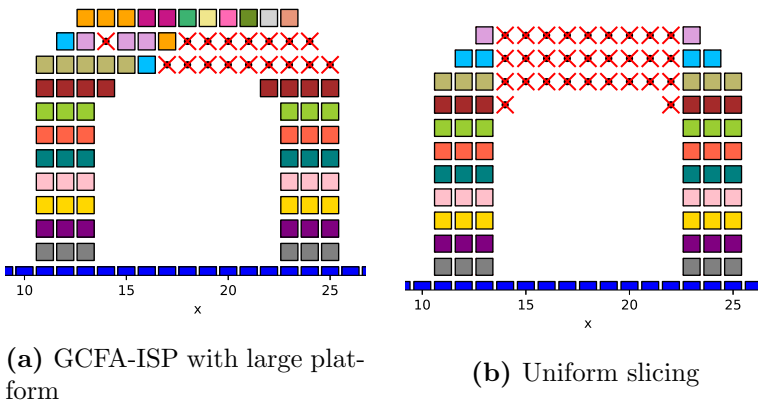


Figure 6.6: Object 5 – The worst result with new method vs. uniform slicing

Table 6.6: Comparing number of missed voxels of the worst result obtained using the new method and uniform slicing.

Method	Missed voxels		
	CFA	Uniform	
<i>Object</i>			
1	16	29	Fig. 6.2
2	3	18	Fig. 6.3
3	6	32	Fig. 6.4
4	0	14	Fig. 6.5
5	17	29	Fig. 6.6

6.4 Hardware

Many types of 6 DOF systems exist. In this Section three different configurations will be compared and discussed. First, the benefits and weaknesses of the hardware used in the experiments of Dai et al. (2018), presented here in Section 4.5, will be discussed. This is followed by an evaluation of the inverted system, where the platform is fixed instead of the deposition tool. To conclude, a new configuration, based on the two successive systems, is proposed.

6.4.1 Fixed nozzle

Figure 6.7 show the system from Section 4.5. The authors of Dai et al. (2018) argue that this configuration benefits the realized object because the filament adhesion is helped by gravity, which due to the fixed deposition tool is always aligned with the build direction.

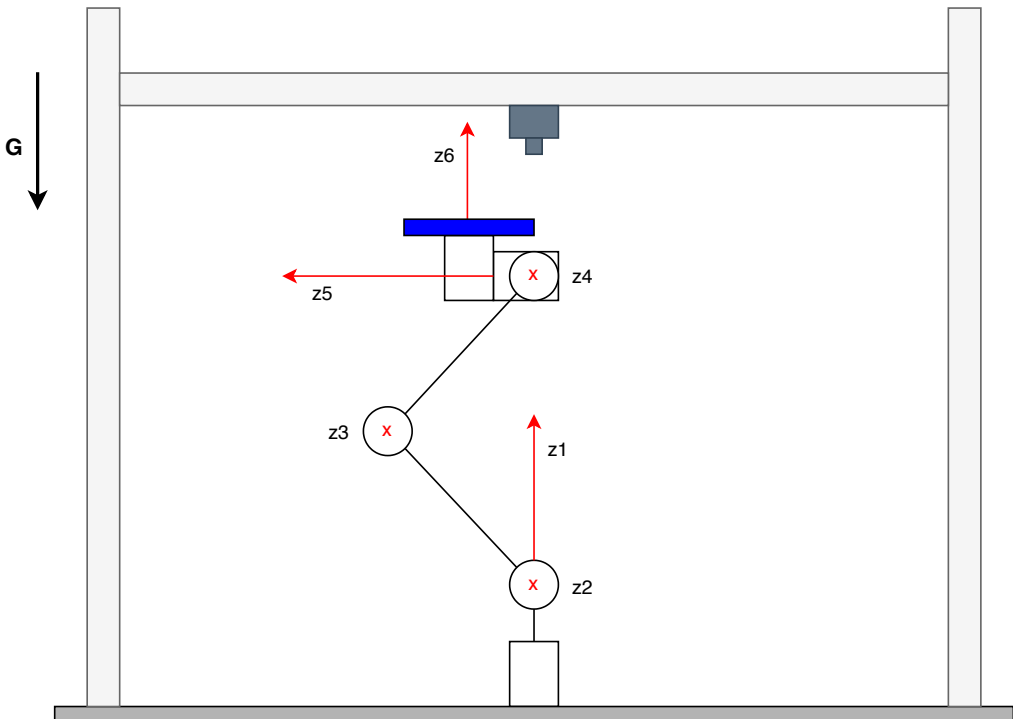


Figure 6.7: Hardware where the platform is attached to the wrist of the robot arm and the manufacturing tool is attached to an overhead beam.

Another important property to notice is that the platform can be chosen small

in this case. Since the platform size often affect how the individual layers are generated, this is a strength to this configuration.

A weakness of this system is that it cannot be used to manufacture a general object. If the object is too heavy, the robot arm may not have enough power to lift it or to move it with sufficient accuracy. In addition to the weight-constraint, the object must also be small enough to fit inside the frames holding the nozzle, similar to using the gantry configuration.

6.4.2 Fixed platform

For most robotic AM systems proposed in research, the configuration in Figure 6.8 is the most commonly proposed. It is popular because it enables the manufacturing of large scale objects.

A drawback in using this configuration when realizing objects using the method proposed in Dai et al. (2018), presented in Chapter 4, is that the working platform used in calculating the accessible surface from Definition 4 in Section 4.4, in reality, is much larger than the object to be realized. This is because the object, in general, is located on the ground, and for the convex front to in reality be the conservative accessible surface, the platform must be set as the ground.

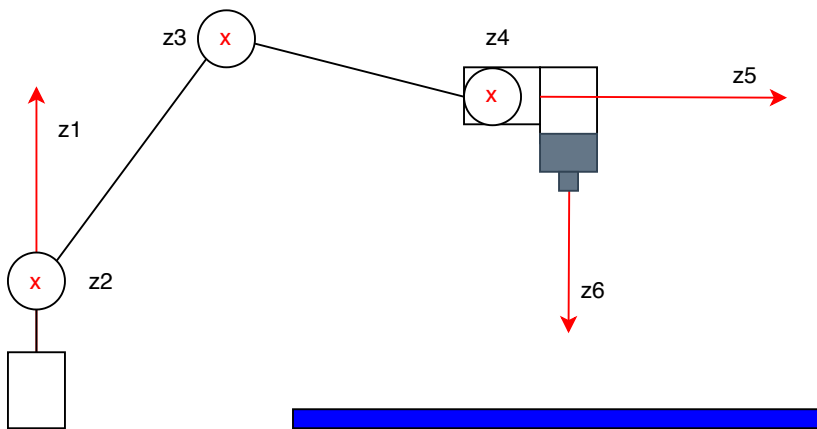


Figure 6.8: Hardware where the platform is on the floor and the manufacturing tool is attached to the wrist of the robot arm.

6.4.3 Fixed, elevated platform

Based on the drawbacks of the hardware discussed in Section 6.4.1 and Section 6.4.2, this thesis propose a new and improved hardware for the support-free manufacturing of general models. By utilizing the small platform from the system in 6.4.1 and the ability of manufacturing large scale objects from the system in Section 6.4.2 we get the new and improved system in Figure 6.9.

In this system, the platform is elevated from the ground so that it can be defined as small when computing the convex front in the algorithms. The robot arm is kept from the system in Figure 6.8 so that the object size is not constrained by the size of the frame holding the deposition tool. The elevation system lifting the platform can be constructed so that it can hold the weight of the object better than the arm.

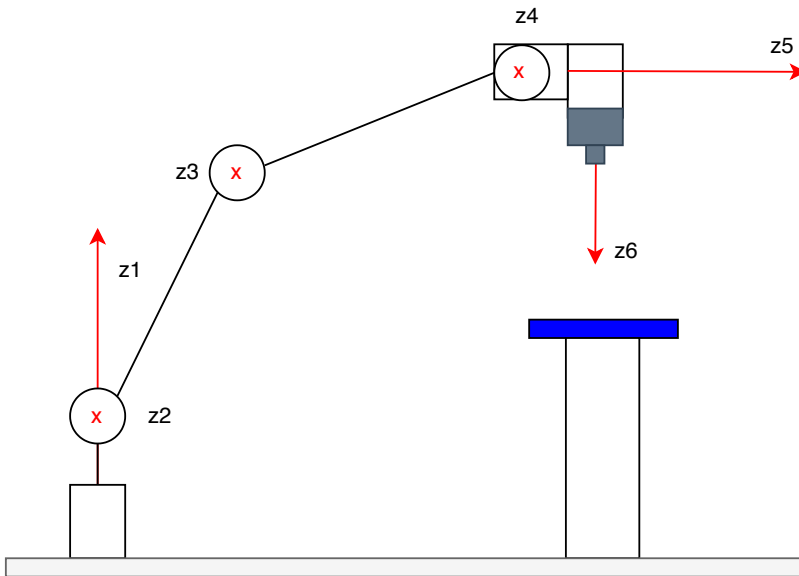


Figure 6.9: Hardware where the platform is attached to an elevation-device and the manufacturing tool is attached to the wrist of the robot arm.

6.4.4 Summary

Table 6.7 summarizes the above mentioned benefits of each hardware.

Table 6.7: Overview of which hardware obtains what properties.

<i>Hardware property</i>	Hardware		
	Fig. 6.7	Fig. 6.8	Fig. 6.9
Large scale manufacturing	×	✓	✓
Enables for use of small working platform	✓	✓	×
Is helped by gravity	✓	×	×

To best benefit from the new system proposed in Dai et al. (2018), the working platform should be as small as possible. For large scale manufacturing, the deposition-tool beam and the wrist-attached platform is a huge disadvantage, so system 2 or 3 should be chosen in this case.

Chapter 7

Conclusion and future work

7.1 Conclusion

In this thesis, two methods for decomposing arbitrary objects into manufacturing layers, proposed in Dai et al. (2018) [3], have been implemented and tested. Different comparisons were made of the results and new hardware was proposed based on the test results and the comparisons.

The methods implemented were referred to as GCFA and GCFA-ISP and both output what was referred to as a growing field, a list containing a sequence of manufacturing layers for AM. The implementations were made based on two algorithms from the supplementary documents of the paper [54]. Changes were made to both algorithms in order to produce satisfying simulation results.

The methods took a voxel-representation of an object as input and accumulated the voxels into specific manufacturing layers which together made up the growing field. For this task, the GCFA method used a greedy approach. The GCFA-ISP used a greedy approach to propose a layer and when this layer produced shadowed voxels, applied an incremental one-step look-ahead approach to try to improve the layer by making it shadow fewer voxels.

The methods were implemented using the Python programming language. Before testing, suitable test objects were created in order to efficiently reveal strengths and weaknesses of the methods. Each test object was tested together with three different platform sizes. The results, i.e. the growing fields, were visualized as layer plots where each layer was given a unique color. Comparisons of each method on the same (platform, object)-pair were performed visually and

numerically. In the end, the worst result generated by either the GCFA or the GCFA-ISP method for each object was compared to the same object decomposed by a common AM fabrication slicing method.

The results showed that in most cases the GCFA-ISP method did generate a better result compared to the GCFA method. In the cases it did not improve the result it was either because the choice of \mathcal{Q} , the ordering of the voxels in a layer when checked for causing shadowed voxels, or the object had a particularly difficult configuration that caused the method to fail.

The results also showed that the platform size impacts the generation of the layers. It was shown that because a large platform increases the size of the conservative accessible surface, more voxels will be situated inside the region and thus more voxels will end up shadowed.

For all objects, it was verified that fewer voxels were missed when the object was decomposed into curved layers than into planar layers. Uniform, planar layer decomposition methods are dependent on the use of support-structures in order for an object to be directly manufactured.

To conclude, the methods investigated in this thesis, by decomposing an object into curved layers, do utilize the abilities for a robot to specify both position and orientation of the manufacturing tool, to realize object of arbitrary size and shape as long as the object can be situated on an isolated, or elevated platform for manufacturing.

7.2 Future work

In this section, topics for further work is proposed. We start by revisiting the pipeline of the multi-axis support-free manufacturing robotic AM system [3] from Section 4.2:

1. Obtain digital object
2. Object sampling
3. Divide the object into manufacturing layers
4. Calculate tool paths to cover each manufacturing layer
5. Translate tool paths into machine code
6. Realize object

We will now have a look at the steps individually and propose future work on each step. Some methods have been developed by the author in order to solve the tasks of step 1 and step 2, but work on these areas has not been emphasized. Therefore, methods for generating and sampling digital models are proposed for further work. Step 1,5 and 6 have been out of scope for this thesis and is also suggested for further work.

Obtain digital model

A wide variety of methods and already existing software may be used to solve this task. Software such as FreeCAD mentioned in Chapter 2 and similar programs may be used for this task. The reason why this has not been investigated further in this thesis is because of the background and interests of the authors, and that it was not necessary to have an automated system for the generation of objects to perform the tasks of the scope of the thesis.

It is, however, a necessary step in order to develop a complete framework for the robotic AM system of concern and should, for this reason, be investigated further.

Sampling of object

The sampling of the object can be viewed as the connector between a CAD software and the AM system investigated in this thesis. In many well established AM systems, the input to the object processing algorithms such as slicers and path generators is, as reviewed in Section 2.1.5, a tessellated object, i.e. triangular approximated version of the CAD model. In this system, however, the input is a list of the coordinates of each point of the sampled object.

Therefore methods for transforming a CAD model into points must be reviewed and tested, or developed, for this application.

Layer generation

The paper propose 4 methods for the generation of layers. This thesis has focused on implementing two of them (GCFA, GCFA-ISP). The other methods should be investigated and implemented for evaluation and comparison purposes. Other methods, influenced by the methods in the paper, may also be developed.

Choices of Q in the GCFA-ISP method should be investigated further and more 3D objects should be tested.

Tool-path calculation

The paper investigated in this thesis does propose a method for calculating tool-paths to cover curved layers. Complete algorithms for this task is however not presented together with the paper.

Further work on this area can be to either study the method proposed in the paper using Fermat spirals and geodesic metrics or to look at other methods for connecting the points of each layer in the growing fields.

Translate tool-paths

After the tool paths have been calculated, they must be sent to run on some hardware.

The system investigated have been developed for the hardware from Section 4.5. An illustration of this hardware can be found in Figure 4.25. In order to use this system to manufacture objects of large scale, application for an inverted system, see Figure 2.10, is desirable. An investigation must be made to find out if the output of the system is not suited for this system, or if changes must be made to run it on this hardware.

It is the author's together with supervisors' conclusion that this is a possible task. If the output of the tool-path program is an ordered list of coordinates, investigations must be made to find out if this path can be realized by both systems.

7.2.1 Physical experiments

To investigate and evaluate the result of the system in total, experiments must be performed and are left for further work as no program for generating tool-paths has been implemented during the work with this thesis.

Appendix A

Mathematical formulas

Definition 7. The n -norm is defined as

$$\|\mathbf{x}\|_n = \sqrt[n]{\sum_{i=1}^k |x_i|^n} \quad p \geq 1$$

Definition 8. The convex hull of a set of points S in n dimensions is the intersection of all convex sets containing S . For N points p_1, \dots, p_N , the convex hull C is given by the expression

$$C = \left\{ \sum_{j=1}^N \lambda_j p_j : \lambda_j \geq 0 \forall j \quad \text{and} \quad \sum_{j=1}^N \lambda_j = 1 \right\}$$

The convex hull of a set of points is the smallest convex set that contains the points [50]. In three dimensions, the convex hull is a convex polyhedron of a finite amount of points [51]. A polyhedron consists of n polygonal faces interconnected by $n + 1$ vertices. A convex polyhedron is made from a convex set of points.

Definition 9. (*Convex set*) A set S is convex if for all x and y in S , the line segment connecting x and y is included in S .)

Appendix B

Algorithms

The following algorithms can also be found as Algorithm 1 and Algorithm 2 in the supplementary document [54] of [3].

Algorithm 3 GreedyGrowingCFA

Input: Voxel representation of a solid model, $\bar{\mathcal{H}} = \{v_{i,j,k}\}$

Output: A growing field $\mathcal{G}(\cdot)$ with value defined on every voxel of $\bar{\mathcal{H}}$

- 1: Adding all voxels adjacent to the platform \mathcal{T} to the first layer \mathcal{L}_1 , as a set of voxels;
 - 2: Set \mathcal{L}_1 as the current working layer \mathcal{L}_c and $\mathcal{C}_{prev} = \emptyset$;
 - 3: Set the layer index $\tau = 1$;
 - 4: **while** $\mathcal{L}_c \neq \emptyset$ **do**
 - 5: Add all voxels of \mathcal{L}_c into the already processed set, \mathcal{V} ;
 - 6: Compute the new convex-front by the convex hull of \mathcal{C}_{prev} , \mathcal{L}_c and \mathcal{T} as $\mathcal{C}_c = \mathcal{C}(\mathcal{C}_{prev} \cup \mathcal{L}_c \cup \mathcal{T})$;
 - 7: Set $\mathcal{L}_{next} = \emptyset$ and $\tau = \tau + 1$;
 - 8: **for** each $v_{i,j,k} \in \mathcal{L}_c$ **do**
 - 9: **for** each $v_{r,s,t} \in \mathcal{N}(v_{i,j,k})$ **do**
 - 10: **if** $v_{r,s,t}$ *NOT inside* \mathcal{C}_c **then**
 - 11: **if** $v_{r,s,t} \notin \mathcal{V}$ *AND* $v_{r,s,t} \notin \mathcal{L}_{next}$ **then**
 - 12: Add $v_{r,s,t}$ into \mathcal{L}_{next} ;
 - 13: **end if**
 - 14: **end if**
 - 15: **end for**
 - 16: **end for**
 - 17: **for** each $v_{r,s,t} \in \mathcal{L}_{next}$ **do**
 - 18: Assign the field-value as $\mathcal{G}(\mathbf{c}(v_{r,s,t})) = \tau$
 - 19: **end for**
 - 20: Set $\mathcal{L}_c = \mathcal{L}_{next}$ and $\mathcal{C}_{prev} = \mathcal{C}_c$
 - 21: **end while**=0
-

Algorithm 4 IncrementalShadowPrevention

Input: The voxel set of an input model $\bar{\mathcal{H}}$, the set of processed voxels \mathcal{V} , the next layer \mathcal{L}_{next} and the current set of shadow region \mathcal{S}_c

Output: An reduced set of \mathcal{L}_{next}

- 1: Set $\mathcal{S}_p = \emptyset$, and $\tilde{\mathcal{L}} = \emptyset$;
 - 2: Compute $\mathcal{C}_p = \mathcal{C}(\mathcal{C}_c \cup \mathcal{T} \cup \mathcal{L}_{next})$;
 - 3: $\forall v \in (\bar{\mathcal{H}} \setminus \mathcal{V})$, add v into \mathcal{S}_p if it is *inside* \mathcal{C}_p ;
 - 4: **if** $\mathcal{S}_p = \mathcal{S}_c$ **then**
 - 5: Return \mathcal{L}_{next}
 - 6: **end if**
 - 7: Determine a heuristic sequence \mathcal{Q} of voxels in \mathcal{L}_{next} by a flooding algorithm;
 - 8: **while** $\mathcal{Q} \neq \emptyset$ **do**
 - 9: Remove a voxel from v from the head of \mathcal{Q} ;
 - 10: Compute the set of shadowed voxels \mathcal{S}_t according to $\mathcal{C}_t = \mathcal{C}(\mathcal{C}_c \cup \mathcal{T} \cup \tilde{\mathcal{L}} \cup v)$;
 - 11: Add v into $\tilde{\mathcal{L}}$ if $\mathcal{S}_t = \mathcal{S}_c$;
 - 12: **end while**
 - 13: **if** $\tilde{\mathcal{L}} \neq \emptyset$ **then**
 - 14: Set $\mathcal{L}_{next} = \tilde{\mathcal{L}}$
 - 15: **else** Set $\mathcal{S}_c = \mathcal{S}_p$;
 - 16: **end if**
 - 17: **return** \mathcal{L}_{next}
-

Appendix C

Code

The implemented algorithms can be found in attached documents of this thesis. The name of the project is "Robotic AM system". These are the implemented modules of the program:

- fileprocessing.py
- utilities.py
- convex_front_advancing.py
- plot2d.py
- plot3d.py
- main.py

The object can be found in a separate folder-hierarchy inside the project folder.:

- Objects
 - 3D
 - 2D

Bibliography

- [1] “SFI manufacturing.” Available: <https://www.sfimanufacturing.no/> [accessed 12.12.19].
- [2] “Global wealt report 2019: global wealth rises by 2.6% driven by us chine, despite trade tensions.” Available: <https://www.credit-suisse.com/about-us-news/en/articles/media-releases/global-wealth-report-2019-global-wealth-rises-by-2-6-driven-by-201910.html> [accessed 18.12.19].
- [3] C. Dai, C. C. L. Wang, C. Wu, S. Lefebvre, G. Fang, and Y. Liu, “Support-free volume printing by multi-axis motion,” *ACM Transactions on Graphics*, vol. 37, August 2018.
- [4] H. Boge, “A literature review on the use of robot technology in additive manufacturing,” May 2019.
- [5] *Standard Terminology for Additive Manufacturing – General Principles – Terminology*. International Organization for Standardization, ISO/ASTM 52900:2015(E), 2015.
- [6] H. Kodama, “Automatic method for fabricating a three-dimensional plastic model with photo-hardening polymer,” *Review of Scientific Instruments*, vol. 52, pp. 1770 – 1773, December 1981.
- [7] D. Chakraborty, B. A. Reddy, and A. R. Choudhury, “Extruder path generation for cruved layer fused deposition modeling,” *Computer-Aided Design*, vol. 40, pp. 235–243, 2008.
- [8] T. Llewellyn-Jones, R. Allen, and R. Trask, “Curved layer fused filament fabrication using automated toolpath generation,” *3D printing and Additive Manufacturing*, vol. 3, pp. 236–243, 2016.
- [9] M. L. Griffith, “Free form fabrication of metallic components using laser enigneered net shaping (lens,tm),” *Time-compression technologies*, sept 1996.

- [10] K. M. Taminger and R. A. Hafley, “Electron beam freeform fabrication for cost effective near-net shape manufacturing,” *NASA Langley Research Center*, 2006.
- [11] S. Ashley, “Rapid Prototyping Systems,” *Mechanical engineering*, vol. 113, pp. 34–43, 1999.
- [12] W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C. B. Williams, C. C. L. Wang, Y. C. Shin, S. Zhang, and P. D. Zavattieri, “The status, challenges, and future of additive manufacturing in engineering,” *Computer-Aided Design*, vol. 69, pp. 65–89, 2015.
- [13] C. Weller, R. Kleer, and F. T. Piller, “Economic implications of 3D printing: Market structure models in light of additive manufacturing revisited,” *International Journal of Production Economics*, vol. 164, pp. 43–56, March 2015.
- [14] I. Gibson, D. W. Rosen, and B. Stucker, *Additive Manufacturing Technologies*. New York: Springer Science+Business Media, 2010.
- [15] H. Kim, Y. Lin, and T. B. Tseng, “A review on quality control in additive manufacturing,” *Rapid Prototyping Journal*, vol. 24, pp. 645–669, 2018.
- [16] J. P. Kruth, “Material in-process manufacturing by rapid prototyping techniques,” *CIRP Annals*, vol. 40, pp. 603 – 614, February 1991.
- [17] C. B. Williams, F. Mistree, and D. W. Rosen, “A functional classification framework for the conceptual design of additive manufacturing technologies,” *Journal of Mechanical Design*, vol. 133, dec 2011.
- [18] M. Livesu, D. Ellero, J. Martínez, S. Lefebvre, and M. Attene, “From 3D models to 3D prints: an overview of the processing pipeline,” *Computer Graphics Forum*, vol. 36, pp. 573–564, May 2017.
- [19] R. Vaidyanathan, “Additive manufacturing technologies for polymers and composites,” in *Additive Manufacturing*. A. Bandyopadhyay and S. Bose, Ed. Boca Raton: CRC Press, Taylor & Francis Group, 2016, pp. 19–64.
- [20] S. S. Crump, “Apparatus and method for creating three-dimensional objects.” U.S patent 5.121.329, 1989.
- [21] “Fused filament fabrication.” Available: https://en.wikipedia.org/wiki/Fused_filament_fabrication [accessed 15.04.19].

- [22] Z. Quan, A. Wu, M. Keefe, X. Qin, J. Yu, J. Suhr, J. Byun, B. Kim, and T. Chou, "Additive manufacturing of multi-directional preforms for composites: opportunities and challenges," *Materials Today*, vol. 18, pp. 503–512, November 2015.
- [23] "Cartesian coordinate system." Available: https://en.wikipedia.org/wiki/Cartesian_coordinate_system [accessed 08.11.19].
- [24] S. Keating and N. Oxman, "Compound fabrication: A multi-functional robotic platform for digital design and fabrication," *Robotics and Computer-Integrated Manufacturing*, vol. 29, pp. 439–448, May 2013.
- [25] K. L. Narayan, K. M. Rao, and M. M. M. Sarcara, *Computer Aided Design and Manufacturing*. New Delhi: Prentice-Hall of India, 2008.
- [26] Wikipedia, "Tessellation (computer graphics)." Available: [https://en.wikipedia.org/wiki/Tessellation_\(computer_graphics\)](https://en.wikipedia.org/wiki/Tessellation_(computer_graphics)) [Accessed 16.04.19].
- [27] A. Hällgren, L. Pejryd, and J. Ekengren, "3d data export for additive manufacturing - improving geometric accuracy," *Procedia CIRP*, vol. 50, pp. 518–523, 12 2016.
- [28] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. US: John Wiley & sons inc., 2006.
- [29] "Robot end effector." Available: https://en.wikipedia.org/wiki/Robot_end_effector [accessed 04.05.19].
- [30] S. J. Keating, J. C. Leland, L. Cai, and N. Oxman, "Toward site-specific and self-sufficient robotic fabrication on architectural scales," *Science Robotics*, vol. 2, April 2017.
- [31] S. Lim, R. A. Buswell, T. T. Le, S. A. Austin, A. G. F. Gibb, and T. Thorpe, "Developments in construction-scale additive manufacturing processes," *Automation in Construction*, vol. 21, pp. 262–268, June 2012.
- [32] E. Barnett and C. Gosselin, "Large-scale 3D printing with a cable-suspended robot," *Additive Manufacturing*, vol. 7, pp. 27–44, May 2015.
- [33] K. Hu, S. Jin, and C. C. L. Wang, "Support slimming for single material based additive manufacturing," *Computer-Aided Design*, vol. 65, pp. 1–10, 2015.

- [34] I. Demir, D. F. Aliaga, and B. Benes, “Near-convex decomposition and layering for efficient 3D printing,” *Additive manufacturing*, vol. 21, pp. 383–394, 2018.
- [35] A. Bellini and S. Güceri, “Mechanical characterization of parts fabricated using fused deposition modeling,” *Rapid Prototyping Journal*, vol. 9, pp. 252–264, 2003.
- [36] O. Shkundalova, A. Rimkus, and V. Gribniak, “Structural application of 3D printing technologies: Mechanical properties of printed polymetric material,” *Science – Future of Lithuania*, vol. 10, pp. 1–8, November 2018.
- [37] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Mech, “Stress relief: Improving structural strength of 3D printable objects,” *ACM Transactions on Graphics*, vol. 31, July 2012.
- [38] S. Singamneni, O. Diegel, B. Huang, I. Gibson, and R. Chowdhury, “Curved layer fused deposition modeling,” *Sabinet*, vol. 8, pp. 95–107, 2010.
- [39] C. Wu, C. Dai, G. Fang, Y. Liu, and C. C. L. Wang, “General support-effective decomposition for multi-directional 3D printing,”
- [40] P. Singh and D. Dutta, “Multi-direction slicing for layered manufacturing,” *Journal of Computing and Information Science in Engineering*, vol. 1, pp. 129–142, March 2001.
- [41] X. Zhang, M. Li, J. H. Lim, Y. Weng, Y. W. D. Tay, H. Pham, and Q. Pham, “Large-scale 3d printing by a team of mobile robots,” *Automation in Construction*, vol. 95, pp. 98–106, 2018.
- [42] L. D. Evjemo, S. Moe, J. T. Gravdahl, O. Roulet-Dubonnet, L. T. Gellein, and V. Brøtan, “Additive manufacturing by robot manipulator: An overview of the state of the art and proof-of-concept results,” *22nd IEEE International Conference on Emerging Technologies And Factory Automation*, 2017.
- [43] L. D. Evjemo, G. Langelandsvik, and J. T. Gravdahl, “Wire arc additive manufacturing by robot manipulator: towards creating complex geometries,” *IFAC-PapersOnLine*, vol. 52, pp. 103–109, 2019.
- [44] “Mesh-mould: Robotically fabricated spatial meshes as concrete formwork and reinforcement,”

- [45] C. Gosselin, R. Duballet, P. Roux, N. Gaudillière, J. Dirrenberger, and P. Morel, “Large-scale 3d printing of ultra-high performance concrete - a new processing route for architects and builders,” *Materials and Design*, vol. 100, pp. 102–109, March 2016.
- [46] N. Hack, W. Lauer, F. Gramazio, and M. Kohler, “Mesh mould: Robotically fabricated metal meshes as concrete formwork and reinforcement,” June 2015.
- [47] C. Wu, C. Dai, G. Fang, Y. Liu, and C. Wang, “RoboFDM: A robotic system for support-free fabrication using FDM,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), May 29 - June 3, 2017, Singapore*, pp. 1175–1180.
- [48] H. Zhao, G. Gu, Q. Huang, Y. Chen, C. Tu, B. Benes, H. Zhang, D. Cohen-Or, and B. Chen, “Connected fermet spirals for layered fabrications,” *ACM Transactions on Graphics*, vol. 35, July 2016.
- [49] P. E. Black, “greedy algorithm,” *Dictionary of algorithms and data structures* [online] Available from: <https://xlinux.nist.gov/dads/HTML/greedyalgo.html> [accessed: 11.10.19].
- [50] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa, “The quickhull algorithm for convex hulls,” *ACM Transactions on Mathematical Software*, vol. 22, pp. 469–483, December 1996.
- [51] “Polyhedron.” Available: <https://en.wikipedia.org/wiki/Polyhedron> [accessed 14.10.19].
- [52] “Qhull.” Available: <https://www.qhull.org> [accessed 16.10.19].
- [53] “What’s an efficient way to find if a point lies in the convex hull of a point cloud?.” Available: <https://stackoverflow.com/questions/16750618/whats-an-efficient-way-to-find-if-a-point-lies-in-the-convex-hull-of-a-point-cl> [accessed 29.11.19].
- [54] C. Dai, C. C. L. Wang, C. Wu, A. Lefebvre, G. Fang, and Y. Liu, “Supplementary document: Support-free printing by multi-axis motion,” *ACM Transactions on Graphics*, vol. 36, July 2018.

