Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Nathaniel Wichmann

Entwicklung eines Software-Frameworks zur Erzeugung und Optimierung von Füllstrukturen für den 3D-Druck

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Nathaniel Wichmann

Entwicklung eines Software-Frameworks zur Erzeugung und Optimierung von Füllstrukturen für den 3D-Druck

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer/in: Prof. Dr. rer. nat. Thomas Lehmann
Zweitprüfer/in : Prof. Dr.-Ing. Günther Gravel

Abgabedatum: 03.01.2020

Zusammenfassung

Nathaniel Wichmann

Thema der Bachelorthesis

Entwicklung eines Software-Frameworks zur Erzeugung und Optimierung von Füllstrukturen für den 3D-Druck

Stichworte

Füllstruktur, Strukturoptimierung, 3D-Druck, Additive Fertigung, FFF, Software-Framework

Kurzzusammenfassung

In dieser Arbeit wird ein Open Source Software-Framework entwickelt, das dreidimensionale Füllstrukturen für den 3D-Druck generiert und anhand mehrerer spezifizierbarer Vorgaben optimieren kann. Als geometrische Grundelemente der Füllstruktur werden werden Quader eingesetzt, welche in mehreren Ebenen progressiv unterteilt werden können, um die Optimierung zu erreichen. Es wird demonstiert, dass der zur Optimierung verwendete Algorithmus Füllstrukturen in allen Raumachsen modifiziert. Resultierende Füllstrukturen weisen darüber hinaus eine hohe räumliche Differenzierung auf.

Nathaniel Wichmann

Title of the paper

Development of a software framework for generation and optimization of 3D print infill structures

Keywords

Infill, Structural Optimization, 3D Printing, Additive Manufacturing, FFF, Software Framework

Abstract

The aim of this thesis is the development of an open-source software framework that generates three-dimensional infill structures for 3D printing. The framework shall further optimize the infill to fit several concurrent specifications. To achieve this goal, rectangular hexahedrons, which can be split along several planes during optimization, are utilized as geometric building blocks for the infill. Results show that the optimization algorithm successfully modifies the infill structure in all dimensions. Furthermore, resulting infill structures exhibit a high level of local differentiation.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Problemstellung

Die Füllstruktur, die von Slicern für den 3D-Druck generiert wird, ist bislang größtenteils homogen. Durch eine bessere und selektive Anpassung an vorgegebene Randbedingungen kann unter anderem Material und Druckzeit eingespart sowie die Wahrscheinlichkeit eines Bauteilversagens gesenkt werden. Ziel dieser Arbeit soll es sein, ein Framework zu konzipieren und prototypisch umzusetzen, welches es ermöglichen soll, eine Softwarerepräsentation einer anpassbaren Füllstruktur zu erzeugen und zu modizifieren. Das Framework soll dann im nächsten Schritt in weitere noch nicht existierende Programme eingebunden werden können und die Grundlage einer optimierten Füllstruktur liefern.

Dazu soll ein Konzept für die modifizierbare Füllstruktur wie auch den Optimierungsalgorithmus entwickelt werden. Die Füllstruktur muss in der Lage sein, verschiedene Eigenschaften und Eigenschaftsverläufe darzustellen, sowie auf Optimierungsvorgaben zu reagieren. Der Programmierer, der das Framework einsetzt, soll darüber hinaus die Ausprägung der Füllstruktur über von ihm definierte Vorgaben innerhalb von in den Grundeinstellungen gesteckten Rahmen beeinflussen können. Eine Programmierschnittstelle soll dies ermöglichen. Ebenso müssen Schnittstellen geschaffen werden, um dem Framework Optimierungsvorgaben mitzuteilen wie auch die Füllstruktur an den Programmkontext des Frameworks weiterzureichen. Es muss letztendlich gewährleistet werden, dass das Framework an verschiedene Anforderungen anpassbar bleibt, die mitunter erst nach Fertigstellung dieser Arbeit ersichtlich werden. Ein hohes Grad an Einstellbarkeit und Modularität des Frameworks ist also in diesem prototypischen Entwicklungsschritt nötig.

2. Einführung

Um die Problemstellung, eine Füllstruktur zu erzeugen, zu verstehen, muss die Technologie, die bei der Umsetzung dieser eingesetzt wird, betrachtet werden. Sowohl 3D-Drucker wie auch die Slicer, die die Werkzeugpfade bzw. Schichtinformationen generieren, werden deshalb zunächst vorgestellt.

2.1. 3D-Druck-Technologien

Im Folgenden werden gängige und für die Arbeit relevante kunststoffbasierte Verfahren kurz vorgestellt. Metallverarbeitende Verfahren werden der Übersichtlichkeit halber hier nicht betrachtet. Die Auslegung der Füllstruktur geschieht vor allem auf die Eigenheiten der FFF-Technologie, welche durch das Replicating Rapid Prototyper-Projekt, kurz RepRap-Projekt[1] sehr gut erforscht und dokumentiert ist. Daneben sind Harz- und Pulververfahren ebenso relevant, nehmen aber eine sekundäre Rolle ein.

2.1.1. Fused Filament Fabrication / Fused Deposition Modelling

Die Fused Deposition Manufacturing (FDM) Technologie wurde 1989 von Crump (1989), Mitbegründer von Stratasys, patentiert und seit 1992 kommerziell eingesetzt[2]. Das Patent ist 2009 abgelaufen, woraufhin sich verschiedene Projekte und Unternehmen der Technologie bedienten, darunter das RepRap-Projekt, welches 2004 unter Prof. Adrian Bowyer gegründet worden ist[3]. Da die Bezeichnung FDM markenrechtlich geschützt ist, wurde die generalisierte Bezeichnung Fused Filament Fabrication (FFF) vom RepRap-Projekt eingeführt und hat sich als offizieller Name der Technologie etabliert (Jones u. a., 2011, S. 177). In dieser Arbeit wird demnach die Bezeichnung FFF verwendet.

[1] https://www.reprap.org/wiki/RepRap
[2] http://www.fundinguniverse.com/company-histories/stratasys-inc-history/
[3] https://all3dp.com/history-of-the-reprap-project/

Im FFF-Verfahren (siehe Abb. 2.1) wird ein auch Filament genannter Thermoplastdraht (A) von einem Extruder mittels eines Vortriebsrades (B) in eine Heizdüse (C) geführt, wo dieser aufgeschmolzen wird. Durch den Extrusionsdruck des nachgeführten Materials wird die Schmelze (D) aus der Düse, dessen Öffnung in der Regel 0,2 bis 1,2 mm beträgt, ausgestoßen und als Extrusionsspuren schichtweise abgelegt (E). Durch die eingeführte Wärme der Schmelze werden bereits solidifizierte Spuren oberflächlich wieder angeschmolzen, sodass Extrusionsspuren miteinander verschweißt werden. Mit einem Verfahrsystems kann der Werkzeugkopf (F) mit der Düse in allen drei Raumachsen über einer Bauplattform (H) bewegt werden. Ein Werkstück (G) wird angefangen von der untersten Ebene schichtweise aufgebaut, wobei die Düse die Konturen des Modells abfährt und der Innenraum entweder komplett oder mit einem parkettierten Muster (Füllstruktur) gefüllt wird.

Einige Maschinen besitzen mehrere Düsen, sodass unter Anderem ein gesondertes Stützmaterial eingesetzt werden kann, um ansonsten unmögliche herzustellende Geometrien zu ermöglichen. Nach dem Fertigungsprozess müssen Werkstücke mechanisch von der Bauplattform entfernt werden. Flexible Plattformen erlauben ein Abschälen, während in der Regel ein Abbrechen von einer steifen Plattform nötig ist.

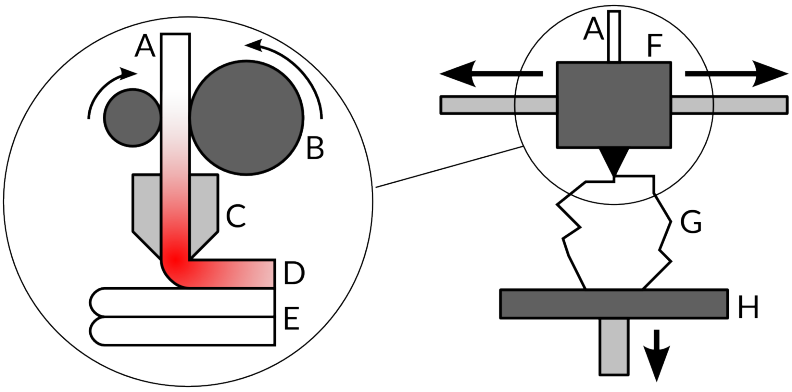


Abbildung 2.1.: Illustration FFF-Verfahren

2.1.2. Harz- und Pulververfahren

Als erste Rapid Prototyping (RP) Technologie wurde die Stereolithographie (SLA) 1984 von Hull (1984), Mitbegründer von 3D Systems, patentiert. Abwandlungen davon werden seit Ablauf des Patents beispielsweise von Formlabs[4] entwickelt. Allen gemein ein flüssiges photosensitives Kunstharz, welches durch energiereiche Strahlung, beispielsweise von einem

[4] https://formlabs.com/blog/introducing-form-3-form-3l-low-force-stereolithography/

Laserstrahl oder Projektor gezielt eingebracht, reagiert und dabei einen duroplastischen Feststoff bildet. Ein Werkstück wird durch das schichtweise Projiziere und Aushärten von Modellquerschnitten hergestellt.

Pulververfahren existieren in verschieden Ausprägungen, wobei diejenigen, die auf Sinterprozessen basieren, für diese Arbeit relevant sind. Selective Laser Sintering (SLS) wurde 1986 von Deckard (1986) patentiert. Dieses Verfahren und Derivate versintern mithilfe eines Laserstrahls schichtweise feines, thermoplastisches Kunststoffpulver zu Modellquerschnitten. Das Werkstück wird während des Prozesses in unversintertes Pulver eingebettet und gestützt, da der Bauraum komplett mit diesem ausgefüllt wird.

2.1.3. Technische Limitation der Druckbarkeit

Die Druckbarkeit von komplexen Modellen wird im Wesentlichen durch folgende Faktoren bestimmt, die verschieden starkes Gewicht bei den vorgestellten Technologien haben. Diese Faktoren sind teilweise gegenseitig bedingt und lassen sich nicht immer sauber trennen. Das FFF-Verfahren wird von einer vergleichsweise großen Community verwendet[5], die zudem rege in der DIY- und Maker-Szene involviert ist. Daher sind für einen Interessierten Informationen zu diesem Verfahren viel einfacher erhältlich, was sich in den folgenden Beschreibungen widerspiegelt.

Überhänge

Überhänge sind Teile des Modells, deren Geometrie in einem kleineren Winkel als 90° zur Bauplattform stehen. Ein stumpfer Winkel von nahezu 90° ist mit jedem Verfahren zu realisieren, doch je spitzer dieser wird, desto schwieriger wird es für das FFF- und das SLA-Verfahren, diese ohne Stützstruktur (siehe Abschnitt 2.1.3) umzusetzen. Pulverbasierte Verfahren sind hier im Vorteil, da vorhandenes Pulver das Modell abstützen. Das FFF-Verfahren dagegen benötigt eine gewisse Oberfläche, auf der die aktuelle Extrusionsspur abgelegt werden kann. Bei spitzen Überhangswinkeln wird diese zunehmend geringer, sodass die Extrusionsspur nicht mehr genügend gestützt wird und sich vom Modell löst. Je geringer die Schichtdicke ist, desto spitzere Winkel sind zu realisieren, da sich die Stützwirkung der vorherigen Schicht umgekehrt proportional zum Überhangswinkel verhält. Deim SLA-Verfahren können stark ausgeprägte Überhänge von dünnen Strukturen beim Anheben der Bauplattform dazu führen, dass diese durch Schälkräfte vom Modell abbrechen[6].

[5] https://www.este.it/images/file-pdf/3D-Printing_Survey2013.pdf
[6] https://support.formlabs.com/s/article/Design-Specs?language=en_US#minimum-unsupported-overhang-angle

Überbrückungen

Aus dem gleichen Grund wie bei Überhängen lassen sich Überbrückungen sehr gut mit pulverbasierten Verfahren herstellen. Überbrückungen sind von mindestens zwei Ankerpunkten unterstützte, annähernd zur Bauplattform parallele Modellgeometrie, die ansonsten keine Anbindung zum restlichen Modell aufweist. Das SLA-Verfahren ist auch hier durch die Schälkräfte zwischen Tankboden und Modell limitiert[7].

Die Fähigkeit, Überbrückungen herzustellen, ist ebenso beim FFF-Verfahren gegeben, jedoch ist sie durch mehr Bedingungen limitiert als bei den anderen Verfahren[8]. Es muss zunächst gewährleistet sein, dass eine Extrusionsspur an zwei Ankerpunkte angelegt werden kann. Die Extrusionsrichtung ist also, wie in Abb. 2.2 zu sehen, auf Parallelen zur Bauplattform begrenzt. Die Extrusion darf nicht gekrümmt sein, da eine gewisse Spannung der Extrusionsspur aufrecht erhalten werden muss, damit die Überbrückung erfolgreich durchgeführt werden kann.

Der noch weiche Kunststoff der Extrusionsspur wird durch die Schwerkraft nach unten gezogen. Dies führt nicht nur dazu, dass die Extrusionsspur mit zunehmender Länge durchhängt und damit von ihrer vorgesehenen Position abweicht, sondern auch, dass die Schmelze der folgenden Extrusionsspuren keinen direkten Kontakt mit ihr haben. Die Verschweißung beider Spuren ist somit höchstens suboptimal[9]. Ferner kann die Extrusion nicht über beliebig lange Strecken erfolgen. Idealerweise sollte diese kurz gehalten werden, da mit wachsender Überbrückungslänge die Druckqualität der betroffenen Geometrie sinkt.

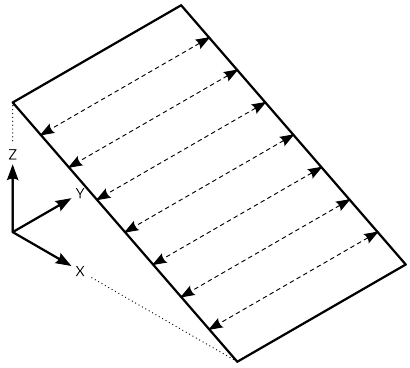


Abbildung 2.2.: Erlaubte Extrusionsrichtung bei schrägen Flächen

[7] https://support.formlabs.com/s/article/Design-Specs?language=en_US#maximum-horizontal-support
[8] https://ultimaker.com/en/resources/19643-bridging
[9] https://www.morgan3dp.com/morgan-bridging-torture-test/

Hohlräume

Bei pulver- und harzbasierten Verfahren verbleibt prozessbedingt zwangsläufig Rohmaterial in völlig umschlossenen Hohlräumen, sodass diese im Modell vermieden werden sollten. Beim SLA kann ein Hohlraum beim Herstellungsprozess durch erzeugten Unterdruck gar zu dessen Versagen führen[10]. Falls diese nicht vermieden werden können, sind Abflusskanäle vorzusehen, die einen Druckausgleich beim SLA[11] sowie das Entfernen des verbliebenen Rohmaterials ermöglichen.

Beim FFF stellen Hohlräume jedoch kein Problem dar. Sie sind dagegen üblich, da in diesem Verfahren eine Füllstruktur erzeugt wird, die das Modell aushöhlt und somit einen geringeren Materialverbrauch bewirkt. Existieren im Modell Hohlräume, müssen die Einschränkungen von Überhängen und Überbrückungen beachtet werden, sodass diese herstellbar sind.

Stützstrukturen

Stützstrukturen sind zusätzlich zum eigentlichen Modell erstellte Geometrien, die dieses beim Herstellungsprozess abstützen sollen. Die oben genannten Herausforderungen durch Überhänge und Überbrückungen können in SLA- und FFF-Verfahren durch Stützstrukturen gelöst werden. Kunststoffbasierte Pulverfahren benötigen dagegen keine Stützstrukturen, da das im Rohmaterial eingebettete Modell durch dieses gestützt wird.

Im SLA bestehen Modell und Stützstrukturen aus dem gleichen Werkstoff, da nur ein Harz im Tank verwendet werden kann. Diese müssen nach dem Aushärten des Werkstückes manuell abgebrochen bzw. -geschnitten und das Modell nach gewünschter Oberflächengüte geschliffen werden. Das FFF-Verfahren kann bei der Verwendung eines einzelnen Extruders Stützstrukturen aus dem gleichen Werkstoff wie das Modell herstellen. Diese werden ähnlich wie beim SLA-Verfahren entfernt. Durch den simultanen Einsatz von mehreren Extrudern können hier Stützstrukturen auch mit einem anderen Werkstoff hergestellt werden als das Modell. Diese sind nahezu spurenlos abbrechbar oder mit Hilfe eines Waschbades auflösbar, was den Aufwand bei der Nachbearbeitung reduziert.

Eine Erstellung von Stützstrukturen durch den Slicer (siehe Abschnitt 2.2) kann in Hohlräumen des Modells geschehen, wenn diese ungünstige Überhänge und Überbrückungen aufweisen. Diese können nachträglich nicht mehr entfernt werden.

[10] https://support.formlabs.com/s/article/Design-Models-for-Printability?language=en_US#resin-drainage

[11] https://support.formlabs.com/s/article/Design-Specs?language=en_US#minimum-drain-hole-diameter

Wandstärke

Das eingesetzte Verfahren bestimmt zu einem Teil erreichbare minimale Wandstärken durch Positioniergenauigkeit und Durchmesser der Heizdüse oder des Laserstrahls bzw. Auflösung des Druckkopfes. Bei Verfahren, die unter Wärmeeinwirkung eine Solidifikation des Modells bewirken, spielt der Wärmetransfer bzw. bei Pulververfahren zusätzlich die Korngröße ebenso eine Rolle.

Die Wandstärke ist nach oben durch Schrumpfung aufgrund von Abkühlung bzw. Aushärtung des Werkstücks begrenzt. Durch sogenanntes "Warping", dem Deformieren des Modells durch ungleichmäßige Abkühlung, sind vor allem günstige FFF-Drucker im Hobbybereich betroffen. Große Modelle sind aufgrund der absoluten Dimensionsänderungen besonders empfindlich[12]. Diesem Problem kann jedoch durch einen beheizten Bauraum und gleichmäßiger, langsamer Abkühlung begegnet werden.

Im FFF-Verfahren ist zudem der Düsendurchmesser ein wichtiges Kriterium für die erreichbare Auflösung. Durch Unter- und Überextrusion kann eine Extrusionsspur erreicht werden, die schmaler bzw. breiter ist als der nominelle Düsendurchmesser. Doch diese Abweichungen sind mit einer eigenen Problematik verbunden, wie Abb. 2.3 zeigt. Bei unveränderter Schichtdicke führt eine Unterextrusion tendenziell zu einer schlechten Verschweißung der Extrusionsspuren sowie zu Hohlräumen. Bei einer Überextrusion besteht die Gefahr, dass überschüssiges Material sich ansammelt, was im schlimmsten Fall durch Kollision mit der Heizdüse zum Versagen des Prozesses oder gar zur Beschädigung der Maschine führt.

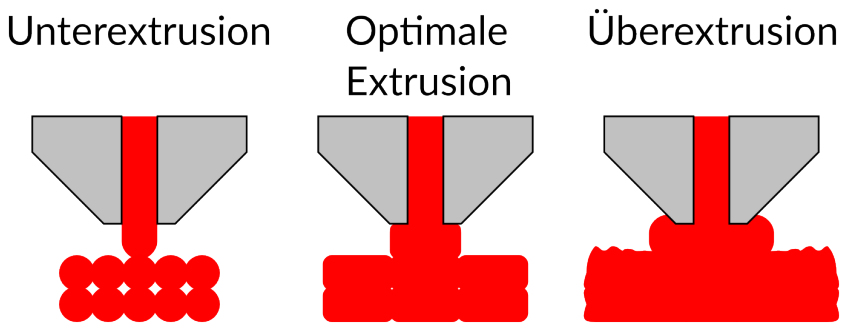


Abbildung 2.3.: Extrusion einer Kunststoffschmelze im FFF-Verfahren. Grau: Düse, Rot: Extrudat

Zwar kann die Problematik durch ein Verändern der Schichtdicke oder dem Verändern des Abstandes der Extrusionsspuren gelöst werden, doch eine optimale Extrusion wird nur dann

[12] https://zortrax.com/blog/reducing-warping-in-3d-prints/

erreicht, wenn die Extrusionsbreite in einem bestimmten Verhältnis zur Schichtdicke steht[13]. Die Wandstärke beim FFF lässt sich also nur in der Extrusionsbreite entsprechenden Inkrementen verändern.

Schichtdicke

Die minimale Schichtdicke variiert zwischen Verfahren. Sie ist zu einem Teil abhängig von der Auflösung des Verfahrsystems der Bauplattform und zum anderen von Materialeigenschaften. Ein dritter, ökonomischer Faktor ist die Herstellungszeit, die umgekehrt proportional zur Schichtdicke ist.

Pulver- und Harzverfahren weisen in der Regel eine geringere minimale Schichtdicke auf als das FFF-Verfahren[14].

Beim FFF hängt die Schichtdicke von Flusseigenschaften der Kunststoffschmelze und dem Düsendurchmesser ab. Sehr feine Schichten lassen sich durch niedrige Extrusionsraten auch bei großen Heizdüsendurchmessern erreichen[15]. Dagegen bedürfen große Schichtdicken einen entsprechenden Durchmesser der Heizdüse, um eine ausreichende Anhaftung an die vorherige Schicht zu gewährleisten (siehe dazu auch Abschnitt 2.1.3).

2.2. Slicer

Um ein 3D-Modell in für den 3D-Drucker lesbare Daten zu konvertieren, wird ein sogenannter Slicer benötigt. Dieser unterteilt das Modell in Scheiben (*Englisch: slices*), da der Aufbau des Werkstücks üblicherweise in Schichten erfolgt (siehe Abb. 2.4).

Bei projektorbasierten SLA-Verfahren wird ein Querschnitt des Modells als Graphik gespeichert, sodass diese zur Belichtung einer Schicht verwendet werden kann. Andere Verfahren wie SLS, FFF und laserbasiertes SLA, die das Werkstück punktuell aufbauen, benötigen dagegen Werkzeugpfade, um den Werkzeugkopf über CNC zu verfahren (siehe Abb. 2.5 und 2.6 für eine beispielhafte Berechnung von Werkzeugpfaden aus einem 3D-Modell, wobei die oberen Schichten in Abb. 2.6 ausgeblendet sind). Dabei muss das Werkstück ausgefüllt werden, was bei SLS und SLA durch ein linienweises Abtasten des Querschnitts erfolgt.

[13] https://blog.prusaprinters.org/everything-about-nozzles-with-a-different-diameter/, Abschnitt "Layer height vs nozzle diameter"
[14] https://www.sculpteo.com/en/glossary/layer-thickness-definition/
[15] http://www.tridimake.com/2013/01/how-fine-can-ultimaker-print.html

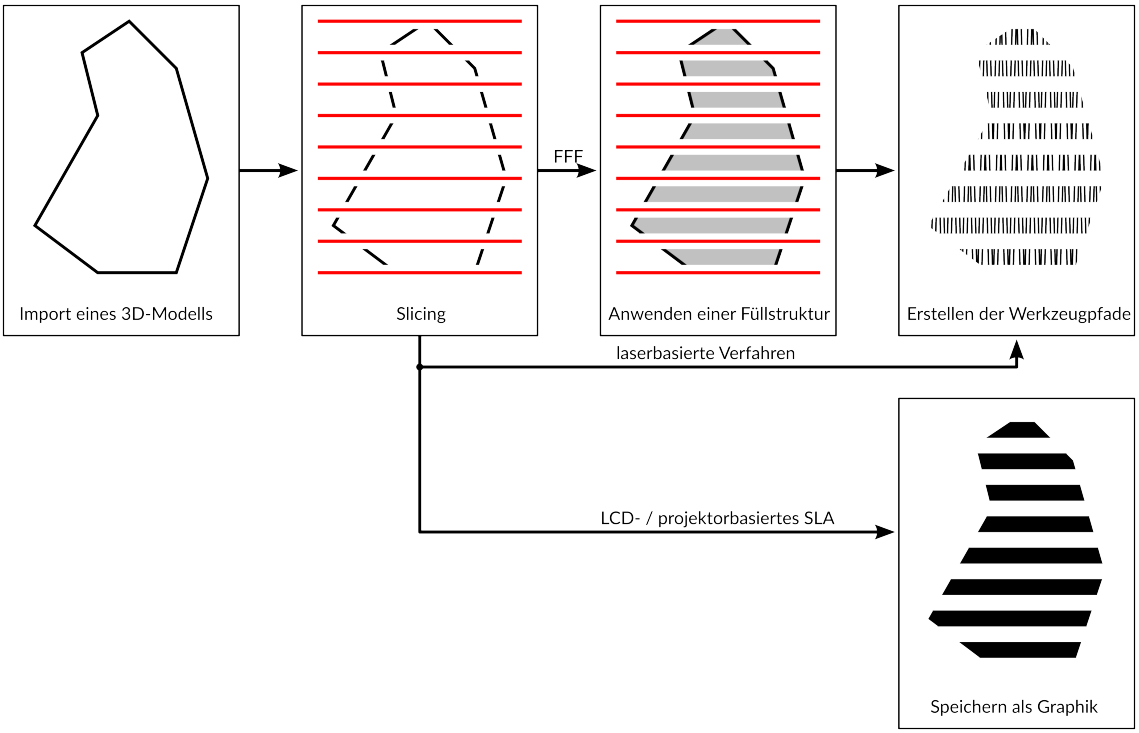


Abbildung 2.4.: Typischer Prozess beim Slicing eines 3D-Modells

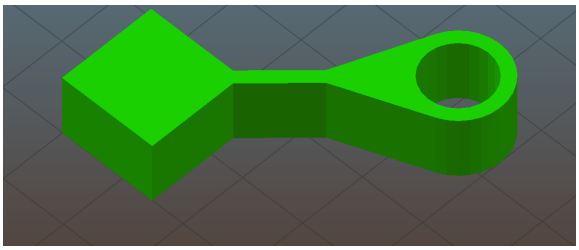


Abbildung 2.5.: In Slic3r 1.2.9 geladenes 3D-Modell

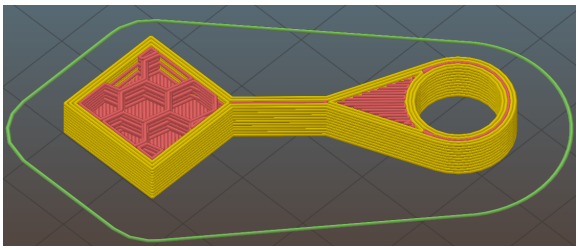


Abbildung 2.6.: Werkzeugpfade, generiert aus 3D-Modell. Gelb: Perimeter (Außenkontur), Rot: Füllstruktur, Grün: Brim (Extrusion, um Arbeitsdruck in der Heizdüse aufzubauen)

2.2.1. Füllstrukturen im Fused Filament Fabrication Verfahren

Auch beim FFF wird eine lineare Füllung verwendet (siehe Detail neben der Bohrung in Abb. 2.6). Wie aus Abb. 2.6 ebenso hervorgeht, können aufgrund der in Abschnitt 2.1.3 diskutierten Eigenheiten des Verfahrens Füllstrukturen (linke Körperhälfte) erzeugt werden. Diese führen zu einer Reduktion im Materialverbrauch sowie zu einer kürzeren Herstellungszeit.

In der Regel wird das Slicing vorgenommen, bevor eine Füllstruktur generiert wird. Dabei werden die Modellquerschnitte ähnlich wie oben beschrieben gefüllt. Es werden jedoch statt einer linearen Füllung (bis auf wenige Ausnahmen) zweidimensionale Muster[16][17] eingesetzt, die zu Hohlräumen im Werkstück führen können (siehe Abb. 2.6 für ein hexagonales Wabenmuster, generiert mit Slic3r 1.2.9).

Die Füllstrukturen können je nach Einstellung unterschiedlich dicht gedruckt werden, sodass die Gesamtdichte des Werkstücks in gewissen Rahmen variiert werden kann. Einstellungen zur Fülldichte wirken global, die Füllstrukturen werden also (mit wenigen Ausnahmen, siehe Abschnitt 3.1) so generiert, dass sie eine regelmäßige Parkettierung einer Schicht erlauben. Dieses Parkettierungsmuster wird über alle Schichten hinweg gleich angesetzt, sodass eine vertikale Extrusion des Grundmusters geschieht, welche in Abb. 2.6 zu erkennen ist.

2.2.2. Füllstrukturen in anderen Verfahren

In der Regel ist es aus in Abschnitt 2.1.3 genannten Gründen unüblich, Füllstrukturen in Harz- und Pulververfahren einzusetzen[18]. Eine Art Füll- oder Gitterstruktur kann dennoch in Harz- und Pulververfahren unter der Stipulation, dass diese wie auch die Außenhülle Durchlässe besitzt, um überschüssiges Rohmaterial abzuleiten, realisiert werden. Diese muss jedoch von einem externen Programm oder dem Designer als Modellgeometrie in das Modell selber integriert werden[19].

2.3. Kontext der Arbeit

Durch die große Designfreiheit in Form und Funktion, die durch die Additive Fertigung gegeben ist, lassen sich Ideen aus der Bionik (z.B. von Lichtenegger (2009) beschrieben) um-

[16] https://manual.slic3r.org/expert-mode/print-settings, Abschnitt ”Infill: Infill Patterns”
[17] https://ultimaker.com/en/resources/52864-infill, Abschnitt ”Infill Pattern”
[18] https://formlabs.com/blog/how-to-hollow-out-3d-models/
[19] https://www.3dsystems.com/sites/default/files/2017-01/3dsystems-sls-designguide-2016.pdf, Abschnitt ”21 - Lattice Structure”

setzen, die mit traditionellen Verfahren nicht möglich waren[20][21]. Siehe auch Liu u. a. (2018). Im Vergleich zu den umformenden und abtragenden Verfahren sind die additiven Verfahren noch recht neu. Zudem ist das Patent zum FFF-Verfahren erst 2009 abgelaufen (siehe Abschnitt 2.1.1), sodass das Forschungsgebiet der Füllstrukturoptimierung vergleichsweise jung ist. Die Strukturoptimierung (bzw. Topologieoptimierung) (Rammerstorfer und Daxner, 2009, S. 43 ff.) des gesamten Körpers auf anliegende Belastung dagegen wird bereits seit längerem auch mit herkömmlichen Fertigungsverfahren betrieben.

2.3.1. Stand der Technik

An dieser Stelle kann eine Unterscheidung getroffen werden, mithilfe derer die Optimierung genauer definiert wird: Durch die traditionelle Strukturoptimierung wird die Macrostruktur (also die generelle Formgebung) des Werkstücks beeinflusst (Klein, 2013, S. 329 ff.), während die Füllstrukturoptimierung in erster Linie die Mesostruktur (kleinere Strukturen im Werkstück) beeinflusst[22].

Die Designfreiheit der Additiven Verfahren erlaubt es, nicht nur die äußere Form eines Werkstücks zu optimieren, sondern auch Füllstrukturen auf Designparameter anzupassen[23]. Die Berücksichtigung der Füllstrukturen eröffnet einen neuen Gestaltungsraum, wofür eigene Designansätze und -richtlinien definiert werden müssen, wie Liu u. a. (2018) beschreiben. Aufgrund der Möglichkeit, bei gleichem Materialverbrauch Körper mit einem größeren Querschnitt herzustellen, erhöhen sich die mechanischen Eigenschaften eines Werkstücks. Beispielsweise werden ein hohes Energieaufnahmevermögen und höhere Beullasten von Groen u. a. (2019) bzw. Clausen u. a. (2017) erkannt. Durch das Schaffen von Hohlräumen, welches alle Füllstrukturen verfolgen, können darüber hinaus thermische und akustische Eigenschaften des Werkstücks modifiziert werden, wie Groen u. a. (2019) feststellen.

[20] https://www.eos.info/industries_markets/aerospace/engines

[21] https://www.eos.info/industries_markets/aerospace/interior

[22] Die Microstruktur (z.B. Extrusionsspuren oder versintertes Pulver) ist bei einigen Verfahren wie dem FFF isotrop und kann ebenfalls durch geeignete Ausrichtung optimiert werden. Deren Optimierung wird jedoch an dieser Stelle nicht weiter behandelt. Siehe dazu Liu u. a. (2018).

[23] Darüber hinaus können mit anderen Verfahren wie PolyJet (https://www.stratasys.com/polyjet-technology) massive Werkstücke hergestellt werden, die aus verschiedenen Materialien bestehen. Diese können durch eine Variation der lokalen Materialzusammensetzung an verschiedene Designparameter angepasst werden (Functionally Graded Materials). Für einen Überblick zur Werkstückoptimierung mithilfe des PolyJet-Verfahrens siehe beispielsweise Arbeiten und Publikationen von Neri Oxman (https://neri.media.mit.edu/neri-oxman.html).

2.3.2. Existierende Arbeiten zur Füllstrukturoptimierung

Jeweils verschiedene Aspekte und Größenordnungen der Füllstruktur werden von den vorgestellten Ansätzen betrachtet. Während Wu u. a. (2018) eine komplett freie Form durch ein lokal volumenlimitiertes Strukturoptimierungsverfahren verfolgen, werden zelluläre Strukturen durch fast alle anderen Ansätze erzeugt. Die gleichzeitige Optimierung der Form und Füllstruktur von Wu u. a. (2017) stellt hierbei einen Sonderfall dar, weil beide Aspekte betrachtet werden.

Der Ansatz von Wu u. a. (2018) erlaubt eine Kontrolle über die Meso- und Makrostruktur, die der von trabekulärer Knochen (siehe auch (Lichtenegger, 2009, S. 4-5)) nachempfunden ist und einen offensichtlich bionischen Ansatz darstellt. Die Füllstruktur wird komplett auf die vermuteten Belastungen optimiert, wobei im Gegensatz zu einer klassischen Strukturoptimierung die Empfindlichkeit auf Schäden und unerwartete Belastungen sehr gering ist, da die Funktion auch bei dem Versagen von einzelnen Strukturelementen aufrecht erhalten werden kann. Die Arbeit zeigt weiterhin, dass plattenartige Strukturen Kräfte besser als offenporige Schäume aufnehmen kann, letztere jedoch in der Natur und beim eingesetzten SLS-Verfahren aufgrund des benötigten Materialdurchflusses benötigt werden. Das FFF-Verfahren besitzt wie in 2.1.3 dargestellt diese Einschränkung nicht. Dieser Ansatz ist in Anbetracht der erreichten Ausprägung der Füllstruktur der komplexeste, weist aber im Vergleich zu Wu u. a. (2016) einen Vorteil in der Steifigkeit des Werkstücks auf. Es wurden kein Vergleich der Berechnungszeit angegeben.

Zelluläre Füllstrukturen dagegen optimieren die Mesostruktur des Werkstücks. Es ist zu bemerken, dass neben dem Ziel, ein ausreichend belastbares Werkstück zu erzeugen, auch andere Optimierungsziele von Wu u. a. (2016) und Podrouzek u. a. (2019) erkannt werden. Podrouzek u. a. (2019) zeigen in ihrer Arbeit, dass die Ausrichtung der Extrusionsspuren im FFF-Verfahren (Microstruktur) eine Isotropie im Werkstück hervorruft, die erheblichen Einfluss auf dessen mechanischen Eigenschaften hat. Dies wird auch von Datenblättern zu Stratasys' FDM-Materialien bestätigt, die eine geringere Belastbarkeit in Z-Richtung attestieren[24]. Podrouzek et al. schlagen weiterhin eine Optimierung der Ausprägung der Füllstruktur als solche vor, damit diese besser auf beliebige Belastungsrichtungen, wie sie bei topologieoptimierten Makrostrukturen vorkommen, reagiert. Durch diesen Ansatz wird die Belastbarkeit des Werkstücks jedoch in der Hauptbelastungsrichtung (Z-Achse) gegenüber extrudierten zweidimensionalen Füllstrukturen gesenkt.

Eine Betrachtung der Strukturen selber wird ebenfalls von Garner u. a. (2018) vorgenommen. Durch die gleichzeitige Berücksichtigung einer Zelle und ihrer Nachbarn in der Füllstruktur

[24] https://www.stratasys.com/-/media/files/material-spec-sheets/mss_fdm_absm30_1117a.pdf

sollen räumliche Eigenschaftsgradienten für Functionally Graded Materials ermöglicht werden. Der Übergang sowie die strukturelle und geometrische Kompabilität von Nachbarzellen rücken hier in den Vordergrund.

Ansätze von Rezaie u. a. (2013), Wu u. a. (2016), Wu (2018), Gopsill u. a. (2019) und Groen u. a. (2019) verfolgen eine Optimierung von zellulären Füllstrukturen. In ihnen wird eine zweidimensionale, extrudierte Struktur dahingehend modifiziert, dass sie Designparameter wie Dichte und mechanische Belastbarkeit erfüllen. Die Füllstruktur weist demnach in der dritten Dimension eine Isotropie auf. Die Herstellbarkeit wird ebenfalls als Randbedingung anerkannt und entsprechend im Design der Füllstruktur berücksichtigt. Dabei werden verschiedene Wege verfolgt. Wu u. a. (2016) schlagen eine durch Unterteilungen optimierbare Struktur vor, deren Wände einen geringsten Überkragungswinkel nicht unterschreiten. Dagegen basiert der Ansatz von Gopsill u. a. (2019) auf einer festen Gitterstruktur, die bei Bedarf mit weiteren Querverstrebungen versehen wird. Die optimierte Struktur nimmt demnach zwei diskreten Zuständen an: unverstärkt und verstärkt. Es ist ebenfalls bemerkenswert, dass dieser Ansatz eine Finite Elemente-Analyse nur als Designgrundlage verwendet und die Struktur nicht iterativ optimiert. Ähnlich gehen Rezaie u. a. (2013) ohne Feedback-Schleife vor, indem das Ergebnis einer klassischen Strukturoptimierung als Grundlage dafür dient, die Dichte von Füllstruktur-Elementen in einem vorgegebenen Gitter zu steuern.

Wu u. a. (2016) wie auch eine spätere Arbeit von Wu (2018) verwenden Baumstrukturen, um eine hierarchisch geordnete Füllstruktur zu erreichen, die durch mehrfache Unterteilungen die nötige Belastbarkeit erreichen. Durch Filterung und Balancierung des verwendeten Quadtrees erreicht Wu (2018) eine höhere Robustheit (geringere Anfälligkeit gegenüber unerwarteten Kräften) der Struktur zu Lasten der maximalen Steifigkeit durch Einschränkung der Füllstruktur-Optimierung[25].

Groen u. a. (2019) verfolgen einen komplett anderen Ansatz, indem sie Zellen in eine Struktur projizieren. Dabei werden diese belastungsgerecht ausgerichtet, sodass ein orthotropes Zellmuster entsteht. Durch den Einsatz eines groben Meshes bei der Strukturoptimierung kann dieser Ansatz mit vergleichsweise geringem Rechenaufwand Füllstrukturen generieren.

Es wird abschließend festgehalten, dass geschlossenporige Schäume den offenporigen zu bevorzugen sind und dass eine zu hohe Optimierung zu einem nachteilhaften Verhalten bei teilweisem Strukturversagen oder unerwarteten Belastungen führt. Eine hohe Robustheit kann durch orthotrope und dadurch weniger optimierte Strukturen erreicht werden.

[25] Vergleiche auch Podrouzek u. a. (2019) und Wu u. a. (2018)

2.3.3. Stand der Technik bei erhältlicher Software

Aktuelle Slicer für das FFF-Verfahren können bereits rudimentäre Optimierungen der Füllstruktur vornehmen, indem der Benutzer Werkstückvolumina Designparameter zuweist[26][27]. Diese werden intuitiv vollzogen und stellen somit keine Grundlage für eine numerisch basierte Optimierung dar. Zudem basieren sie auf extrudierten, zweidimensionalen Strukturen, deren Nachteile ausführlich dargelegt worden sind. Um die Entwicklung eines Slicers voranzutreiben, der die oben dargelegten Ergebnisse verschiedener Autoren umsetzen kann, wird ein Framework zur Optimierung von Füllstrukturen im FFF-Verfahren entwickelt.

[26] https://www.simplify3d.com/support/articles/different-settings-for-different-regions-of-a-model/
[27] https://manual.slic3r.org/advanced/modifier-mesh

3. Analyse

Um die Entwicklung des Frameworks durchzuführen, wird es bei der Anforderungsanalyse zunächst in seine einzelnen Komponenten aufgeteilt. Somit können Schnittstellen und benötigte Ressourcen erkannt und in der Designphase gezielt zusammengeführt werden. Erfasste Anforderungen werden in einer Anforderungsliste (siehe Anhang) zusammengefasst.

3.1. Füllstruktur

Um in Additiven Fertigungsverfahrenerfahren möglichst effizient Bauteile herzustellen, werden mitunter Füllstrukturen eingesetzt (siehe Abschnitt 2.2).

Zwei Hauptkriterien bestimmen während des Druckprozesses die Effizienz, mit der eine Füllstruktur hergestellt werden kann: Herstellungsgeschwindigkeit und Materialverbrauch. Damit verbunden sind weitere Kriterien wie Masse und Belastbarkeit des Bauteils, die vor allem bei dessen Gebrauch eine Rolle spielen. Die Belastbarkeit ist zudem ein Kriterium, welches oftmals konträr zu den anderen wirkt.

Das Ziel dieser Arbeit ist es, einen Prototyp von einer Füllstruktur zu generieren, die sich lokal auf verschiedene Anforderungen anpassen lässt. Prinzipiell kann dies durch Veränderung der Strukturgeometrie und/oder des Werkstoffes geschehen. Beide Wege lassen sich im Prototypenstadium vor allem über das FFF-Verfahren verwirklichen. Die Anpassung der Geometrie lässt sich ebenfalls auf Harz- und Pulververfahren übertragen, mit der in Abschnitt 2.2.2 genannten Einschränkung, dass Öffnungen sowohl in der Füllstruktur als auch in der Außenhülle des Bauteils vorgesehen werden. Aufgrund dieser Faktoren konzentriert sich die Entwicklung der anpassbaren Füllstruktur vor allem auf die Anwendung im FFF-Verfahren. Eine Erweiterbarkeit auf weitere Verfahren sollte dennoch gegeben sein, weshalb eine Modifizierung auf Basis des Füllmaterials hier zunächst nicht in Frage kommt.

3.1.1. Unzulänglichkeit der herkömmlichen Füllstruktur

Herkömmliche Füllstrukturen werden global über das Bauteil berechnet und als zweidimensionales Muster schichtweise in den Modellquerschnitt eingefügt. Zusätzlich wird dieses ver-

tikal extrudiert, wie in Abschnitt 2.2.1 dargestellt. Somit ist bereits eine Isotropie in die Vertikale gegeben (siehe auch Podrouzek u. a. (2019)), welche die Anpassungsfähigkeit in mindestens dieser Achse einschränken würde. Die herkömmlichen Füllstrukturen erfüllen die Anforderung der Anpassbarkeit somit nicht.

3.1.2. Grundlagen für eine verbesserte Füllstruktur

Wie Wu u. a. (2018) bereits demonstriert haben, ist die Berechnung einer idealen Geometrie zur Aufnahme anliegender Lasten durch ein angepasstes Strukturoptimierungsverfahren möglich, aber ressourcenaufwändig. Für diese sind Finite Elemente-Simulationen nötig, was die Anforderungen an zusätzlicher Software steigert. Ebenso ist bereits der erfolgreiche Einsatz von zellulären Strukturen (in zweidimensional extrudierter Ausprägung) für die Anpassung der Füllstruktur durch Rezaie u. a. (2013), Wu u. a. (2016), Wu (2018) und Gopsill u. a. (2019) erwiesen. Wie die Arbeit von Wu u. a. (2016) zusätzlich zeigt, können auf Basis eines solchen vereinfachten Optimierungsansatzes mehrere Eigenschaften, wenn auch separat, modifiziert werden.

Es wird also eine reine Berechnung der idealen Füllstruktur mit einem Strukturoptimierungsverfahren einer Simplifizierung durch Vorgabe von Geometrieelementen durch zelluläre Strukturen gegenübergestellt. Ein Erweitern der zellulären Strukturen kann zu einer wirklichen räumlichen Anpassbarkeit und dem Vermeiden bzw. Reduzieren von ungewünschter Isotropie führen, indem die extrudierten zellulären Strukturen durch dreidimensionale, räumlich angeordnete ersetzt werden. Werden dabei zusätzlich die von Podrouzek u. a. (2019) beschriebenen Designkriterien der Mesostruktur berücksichtigt, kann die Isotropie weiter gesenkt werden.

3.1.3. Füllstruktur basierend auf einem Grundkörper

Da die Ergebnisse von Rezaie u. a. (2013), Wu u. a. (2016), Wu (2018) und Gopsill u. a. (2019) zeigen, dass ein vereinfachter Ansatz durchaus in der Lage ist, eine anpassbare Füllstruktur zu erzeugen, wird auf diesem aufgebaut. Um das oben beschriebene Problem der isotropien Eigenschaftsausprägungen zu reduzieren, wird der Ansatz der räumlichen Parkettierung von Körpern zur Erstellung der Füllstruktur verfolgt. Die zur initialen Parkettierung verwendeten Körper werden nachfolgend Grundkörper genannt, da aus ihnen alle weiteren Ausprägungen hervorgehen. Generalisierend wird immer dann der Begriff Körper verwendet, wenn nicht zwischen Grundkörper und Permutationen dessen differenziert wird.

Über Veränderung bzw. Ersetzen dieser Grundkörper können dann Eigenschaften eingestellt werden. Die spezifische Problematik in diesem Punkt ist es, nicht nur einen bzw. mehrere

geeignete Grundkörper zu finden, die die geforderten Eigenschaften erfüllen, sondern auch, geeignete Modifikationen dieser zu entwickeln, mithilfe derer die lokalen Eigenschaften der Füllstruktur für jeweilige Ansprüche entsprechend angepasst werden können.

3.1.4. Räumliche Parkettierung von Körpern

Um das Problem der räumlichen Parkettierung mit Körpern zu lösen, muss zunächst ihre Geometrie bestimmt werden. Hierzu werden konvexe Polyeder betrachtet. Die Körper müssen im 3D-Raum so angeordnet werden können, dass sie diesen lückenlos und ohne Überlappung füllen. Für diese Anforderung kommt die Untergruppe der raumfüllenden Polyeder in Frage. Unter den Vertretern dieser Gruppe sind der Quader, das dreiseitige sowie das sechsseitige Prisma wohl die bekanntesten. Diese sind auch die einzigen, die durch eine Extrusion aus einer 2D-Repräsentation gewonnen werden können (siehe Abb. 3.1).

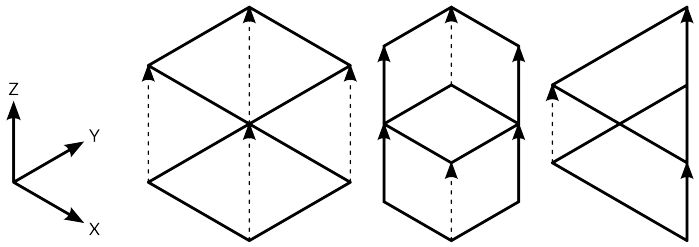


Abbildung 3.1.: Extrusion eines Quadrates, eines Hexagons und eines Dreiecks zum Kubus, sechsseitigen Prisma bzw. dreiseitigen Prisma

Außerdem kommen Parallelipede (darunter die von Wu u. a. (2016) untersuchten extrudierten Rhomben als Spezialfall), rhombische Dodekaeder, Oktaederstümpfe und verdrehte Doppelkeile als einfache Körper in Frage. Nicht-konvexe Polyeder werden nicht betrachtet, da die Form der Grundkörper möglichst simpel gehalten werden soll, um eine einfache Berechnung und Anordnung zu ermöglichen. Aus dem gleichen Grund werden Polyeder mit einer hohen Seitenzahl ausgeblendet.

Eine geringe Isotropie ist bei einer Parkettierung von allen vorgestellten Körpern außer den Prismen und dem verdrehten Doppelkeil gegeben. Diese weisen ein signifikant abweichundes Parkettierungsmuster in Z-Ebene auf, während die anderen Körper in jegliche Richtung gleichmäßig angeordnet werden können. Wie oben erwähnt sind die anschaulichsten Körper hier die in Abb. 3.2 illustrierten Quader, dreiseitige und sechsseitige Prismen. Aus diesen drei ist es nur mit dem Quader möglich, in allen 3 Raumachsen eine gleiche Parkettierung vorzunehmen, wenn dieser zusätzlich die Bedingung erfüllt, ein regelmäßiger Hexaeder (Kubus) zu sein.

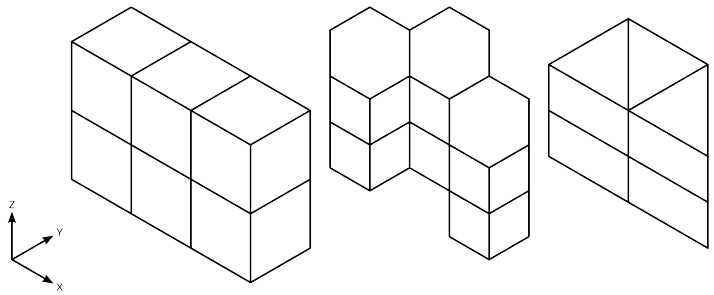


Abbildung 3.2.: Parkettierung mit Hexaedern, sechsseitigen Prismen und dreiseitigen Prismen

Dieser Umstand trifft sich gut mit der weiteren Anforderung, dass die Softwarerepräsentation des Körpers relativ einfach zu implementieren sein sollte. Für den Quader werden nur der Ort des Referenzpunktes des Körpers und seine Maße benötigt, um ihn mitsamt allen seinen Eckpunkten, Kanten und Flächen zu bestimmen, solange seine Kanten zu den Achsen des verwendeten Koordinatensystems ausgerichtet sind (siehe Abb. 3.3). Die Ausnutzung dieses Faktes spart nicht nur Berechnungsaufwand bei der Lokalisierung der Körpergeometrie, sondern auch beim Berechnen des Versatzes zwischen einzelnen Körpern. Körper wie der Oktaederstumpf weisen hier mit steigender Komplexität immer größere Nachteile vor, da sie aufgrund der höheren Anzahl an Eckpunkten sowie der räumlichen Verschachtelung bei der Parkettierung einer aufwändigeren Berechnung bedürfen.

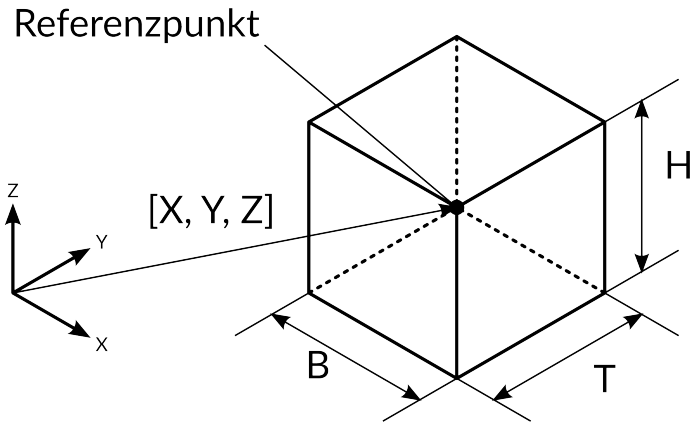


Abbildung 3.3.: Vollständig definierter Hexaeder

3.1.5. Druckbarkeit von Körpern

Weiterhin muss die Druckbarkeit eines Körpers insbesondere im Kontext der FFF-Technologie betrachtet werden, deren Problematik im Abschnitt 2.1.3 genauer erläutert worden ist. Von den drei vorgestellten Körpern eignet sich prinzipiell das dreiseitige Prisma am besten für diese Anforderungen, wenn es mit der kleinsten rechteckigen Fläche parallel zur Bauplattform ausgerichtet wird, da dann zwei von drei Flächen einen Winkel von mindestens 45° zur Bauplattform einnehmen. Zusätzlich wird damit eine nötige Überbrückung minimal gehalten. Allerdings führt ein immer kleinerer Winkel zu einer wachsenden Isotropie.

Der Entwurf eines Grundkörpers sollte diese Umstände berücksichtigen und möglichst kurze Extrusionsspuren in der Luft ermöglichen oder sie wie der rhombische Lösungsansatz von Wu u. a. (2016) gänzlich vermeiden.

3.1.6. Modifikation des Grundkörpers

Wie in Abschnitt 3.1.3 angerissen, können Eigenschaften der Füllstruktur durch Modifikation des Grundkörpers verändert werden. Da Werkstoffeigenschaften selten mit diskret anliegenden Faktoren im Wechselspiel stehen, sondern mit Gradienten (beispielsweise über einen Körper verteilte mechanische Spannungen), muss eine Modifikation ebenfalls Eigenschaftsgradienten annähern können.

Es wird bedacht, dass für ein wahres Kontinuum auch die Körper und deren Geometrie unendlich fein einstellbar sein müssen. Dies ist jedoch technisch nicht gegeben, da jedem Fertigungsverfahren Grenzen gesetzt sind. Gerade das für diese Arbeit gewählte FFF-Verfahren weist bei der erreichbaren Auflösung von Strukturen bedeutende Einschränkungen auf, da die Breite der Extrusionsspuren stark vom Düsendurchmesser abhängen. Zwar kann die Extrusionsbreite in einem gewissem Rahmen moduliert werden, doch dies hat Einflüsse auf mechanische Eigenschaften und den Herstellungsprozess. Siehe hierzu Abschnitt 2.1.3, insbesondere Abb. 2.3. Daher kann ein Kontinuum nur angenähert werden[1].

Die Parkettierung des Modellraumes mit Körpern ist diskreter Natur, die erreichbare Auflösung hängt dabei von den eingesetzten Körpern, deren Dimensionen und den vollzogenen Modifikationen ab. Im Folgenden wird eine nicht abschließende Liste an unterschiedlichen Möglichkeiten beschrieben, um eine geeignete Methode der Modifikation von Körpern zu ermitteln.

[1] Unter Einsatz beispielsweise des viel höher auflösenden SLS-Verfahrens ist diese Annäherung jedoch wesentlich exakter erfolgen.

Parkettierung eines Körpers

Eine Möglichkeit der Modifikation von Eigenschaften der Füllstruktur ist die Parkettierung von Körpern mit weiteren Körpern unterschiedlicher Größe. Indem nach Bedarf mehr Körper und damit mehr Wände eingesetzt werden, können Eigenschaften der Füllstruktur lokal verändert werden (siehe auch Wu u. a. (2016) und Wu (2018)). Das hexagonale Prisma ist hierfür ungeeignet, wenn nur weitere, gleichartige Körper eingesetzt werden sollen, wie Abb. 3.4 im zweidimensionalen Raum zeigt, wohingegen das dreiseitige Prisma und der Quader lückenlos in verschiedenen Größen parkettierbar sind (Abb. 3.5). Wird jedoch wie in Abb. 3.6 die Möglichkeit der Unterteilung in weitere, andersartige Unterkörper betrachtet, ist das hexagonale Prisma durch die höhere Anzahl an Kanten, welche durch Einfügen von Flächen verbunden werden können, im Vorteil, während es unmöglich wird, auf diese Weise das dreiseitige Prisma mit einer ebenen Fläche zu unterteilen.

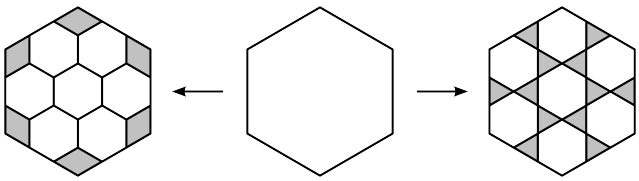


Abbildung 3.4.: Parkettierung eines Sechsecks

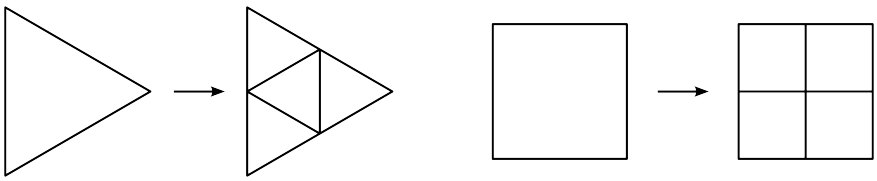


Abbildung 3.5.: Parkettierung eines gleichseitigen Dreiecks und eines Quadrates

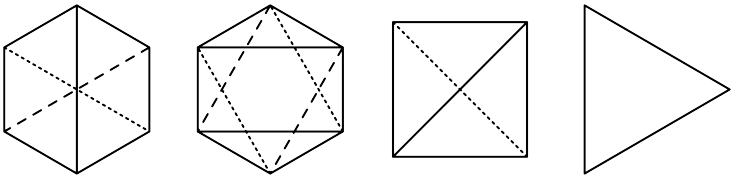


Abbildung 3.6.: Unterteilung von Körpern durch Verbinden von Eckpunkten

Es existieren viele weitere Möglichkeiten der Unterteilung, doch die Umsetzung dieser wird zunehmend komplex, da die Anzahl und Komplexität der nötigen Berechnungen ebenfalls

mit ansteigt. Im Rahmen dieser Arbeit werden also zunächst nur die einfachen Fälle der Unterteilung betrachtet. Diese sind die Halbierung, Viertelung usw. zwischen Mittelpunkten der Kanten bzw. zwischen Eckpunkten, wie in Abb. 3.7 dargestellt. Für eine fortschreitende Unterteilung ist es nötig, dass ein Körper mehrere Symmetrieebenen besitzt, sodass dieser potentiell entlang mehrerer Ebenen unterteilt werden kann. ohne dass die Formeln, die die Unterkörper und deren Parkettierung beschreiben, an Komplexität zunehmen.

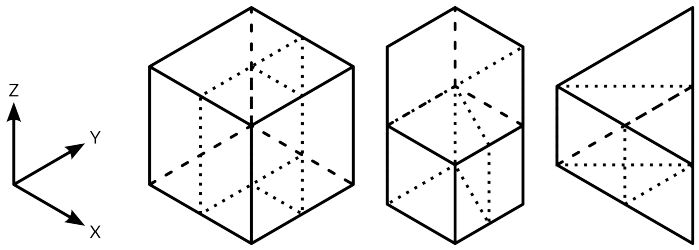


Abbildung 3.7.: Erste Unterteilungen von Körpern durch Halbierung zwischen Mittelpunkten der Kanten bzw. zwischen Eckpunkten.

Verschmelzung von Körpern

Der gegensätzliche Ansatz zur Parkettierung ist die Verschmelzung bzw. Kombination von Körpern. Durch diese ist es möglich, komplexe Körper zu erzeugen (siehe Abb. 3.8). Kombinierte Körper können durch ihre Form besondere Eigenschaften aufweisen, die mit einfachen Körpern nicht zu erreichen sind.

Durch eine höhere Komplexität können beispielsweise Funktionen integriert werden. Compliant Mechanisms[2] wären eine sehr weit entwickelte Anwendung, die durch den gezielten Zusammenschluss einer Region von Körpern verwirklicht werden könnte. Die sogenannte Emergenz[3][4] durch das Manipulieren einer Füllstruktur wird an dieser Stelle nicht weiter betrachtet und soll nur als ein Beispiel der vielfältigen Möglichkeiten einer differenzierten Füllstruktur dienen.

Allerdings bringt die Verschmelzung auch eigene Problemstellungen mit sich. Um Körper miteinander zu kombinieren, muss deren Nachbarschaftsbeziehung bekannt sein, d.h. es muss bekannt sein, ob deren Flächen aneinander grenzen. Siehe hierzu auch Abschnitt 3.1.7.

[2] https://www.compliantmechanisms.byu.edu/
[3] http://consc.net/papers/granada.html
[4] https://www.iep.utm.edu/emergenc/

Die Definition des Referenzpunktes eines Körpers bei einer Verschmelzung muss eindeutig sein. Wird der eines konstituierenden Körpers verwendet, muss eine Entscheidungsregel eingeführt werden, welcher Referenzpunkt gewählt wird. Andererseits kann beispielsweise auch der Schwerpunkt des zusammengesetzten Körpers als Referenzpunkt dienen.

Ein weiteres interessantes Problem ist die mögliche Umschließung eines Körpers wie in Abb. 3.9 dargestellt. Der umschlossene Körper hat keine Verbindung zu anderen Körperbegrenzungen mehr und ist damit nur noch mit Pulververfahren druckbar. Zudem befindet er sich frei im Raum und verliert somit seine Funktion in der Füllstruktur. Dieser Sonderfall muss daher durch besondere Maßnahmen vermieden werden. Von allen hier vorgestellten Modifikationsverfahren weist die Verschmelzung die größten Umsetzungshürden auf, kann aber durch eine hohe erreichbare Komplexität potentiell weitere Funktionalität integrieren, die nur schwer oder unmöglich mit anderen Methoden zu erreichen ist.

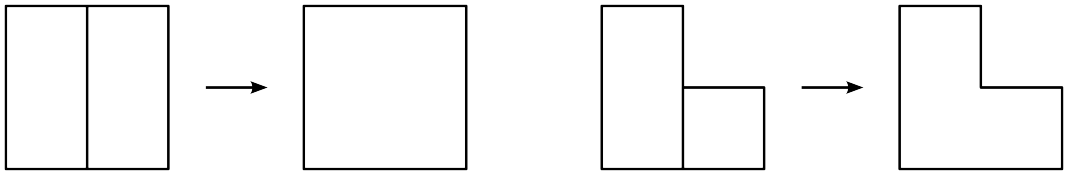


Abbildung 3.8.: Kombination von Körpern. Durch Kombination von verschieden großen Körpern können komplexe Körper gebildet werden.

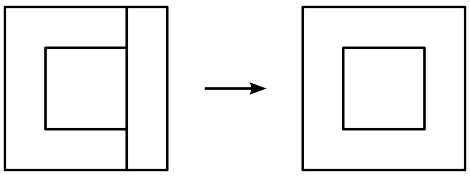


Abbildung 3.9.: Umschließung eines Körpers durch eine Kombination weiterer Körper

Modifikation der Wandstärke

Die Eigenschaften des Grundkörpers können ebenso durch die Variation der Wandstärke verändert werden (siehe auch Rezaie u. a. (2013)). Dieser Ansatz ist für alle Grundkörper geeignet, da nicht auf eine Parkettierung innerhalb des Grundkörpers Rücksicht genommen werden muss. Allerdings muss eine geeignete Größe des Grundkörpers für die initiale Füllstruktur gewählt werden, damit die entstehenden Überbrückungen nicht die technischen

Limitationen des Druckes überschreiten und damit die Modellaußenhülle, die von der Füllstruktur gestützt wird, keine ungewollten Artefakte aufweist (siehe auch Abschnitt 2.1.3).

Werden die Grundkörper zu klein gewählt, kann die Füllstruktur jedoch unter Umständen nicht auf leichtere Ausprägungen hin optimiert werden. Bei einer gewählten mittleren Wandstärke als Anfangswert kann eine Optimierung in beide Richtungen erfolgen. Die Veränderung der Wandstärke muss auf das Innere von Grundkörpern beschränkt werden, da diese aneinandergrenzen und die Wände sich sonst überschneiden würden. Weiterhin setzt sich die Wandstärke aus der von zwei benachbarten Körpern zusammen, wie Abb. 3.10 anhand des Vergleichs zwischen vertikalen und horizontalen Wänden zeigt.

Die Komplexität der Berechnung der Wandstärke ist vom eingesetzten Körper abhängig. Hier ist der Quader wieder im Vorteil, da lediglich die Koordinaten der Eckpunkte entlang der jeweiligen Achsen verschoben werden müssen, während bei anderen Körpern der Winkel, mit dem die Wände des Grundkörpers zueinander stehen, mit in die Berechnung einbezogen werden muss. Abb. 3.10 zeigt eine beispielhafte Progression, in der die Wandstärke mit jedem Schritt verdoppelt wird.

Abbildung 3.10.: Veränderung der Wandstärke des Grundkörpers

Ersetzen des Grundkörpers durch spezialisierte Körper

Ebenso ist denkbar, den Grundkörper durch spezialisierte Körper zu ersetzen, die den jeweiligen Anforderungen am besten angepasst sind. Einen ähnlichen Ansatz verfolgen Gopsill u. a. (2019) mit dem zusätzlichen Einsetzen von Streben in eine vorgegebene Gitterstruktur. Während die oben genannten Methoden darauf basieren, die neue Geometrie aus der gegebenen zu berechnen, kann hier ein vorher berechneter Katalog an Körpern eingesetzt werden, was die Berechnungszeit in der Optimierungsphase verkürzt.

Wie bei der Modifikation der Wandstärke muss die Ausprägung der initialen Füllstruktur gut überlegt sein. Da bei der Ausprägung der spezialisierten Körper fast komplette Gestaltungsfreiheit gegeben ist, kann die Optimierung spezifischer auf die Anforderungen reagieren. Abb. 3.11 zeigt einige beispielhafte Körper, die aus einem Quadrat bzw. dem daraus extrudierten Kubus entstehen könnten.

Allerdings muss schon bei der Konstruktion der Körper die verwendete Technologie bedacht werden. Ein bestimmter Körper kann unter Umständen nicht mit jedem Verfahren umsetzbar, was im Rahmen dieser Arbeit zunächst zu Gunsten des FFF-Verfahrens ignoriert werden kann. Ebenso sind Körper eventuell nicht in jeder Größe umsetzbar. Hier muss das Zusammenspiel zwischen gewählten Grundeinstellungen und möglichen spezialisierten Körpern ausgelotet werden. Ferner steigert ein komplexer Körper den Rechenaufwand beim Slicing. Ein einfacher geometrischer Körper kann mit Hilfe von relativ einfachen mathematischen Formeln beschrieben werden, sodass das Erstellen eines Querschnittes mit wenigen Berechnungen möglich ist. Eine gesteigerte Komplexität bedeutet eine höhere Berechnungszeit, vor allem, wenn unter Umständen nicht mehr auf eine einfache Beschreibung durch Formeln zurückgegriffen werden kann, sondern die Schnittgeraden des Körpers mit der Schnittebene berechnet werden müssen.

Garner u. a. (2018) betrachten darüber hinaus die Nachbarschaft eines Körpers, um die strukturelle Konnektivität der einzelnen Körper zu gewährleisten. Dieser Aspekt muss vor allem bei der Optimierung auf mechanische Eigenschaften berücksichtigt werden. Ebenso muss die Herstellbarkeit gewahrt werden, indem bestimmte Geometriekombinationen ausgeschlossen werden, die zu Defekten in der Füllstruktur führen. Solches kann beispielsweise durch die Definition eines Satzes von Designregeln erreicht werden, die sicherstellen, dass zumindest Klassen von spezialisierten Körpern miteinander kompatibel sind.

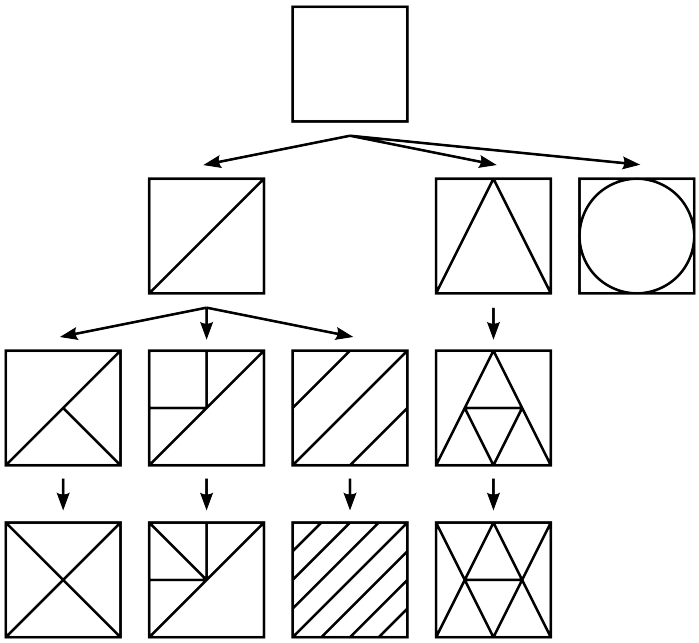


Abbildung 3.11.: Ersetzen des Grundkörpers durch spezialisierte Körper gleicher Außengeometrie

3.1.7. Darstellung der Körper durch eine Datenstruktur

Um die Füllstruktur abzubilden, bedarf es einer geeigneten Datenstruktur, die ihre Eigenheiten und die Zusammenhänge zwischen den einzelnen Körpern darstellen kann. Bei der Verwendung von Grundkörpern stellt sich zwangsläufig eine Art von Gitter ein, wenn mit diesen eine Parkettierung des Raumes vorgenommen wird. Im speziellen Fall der Anpassung durch Unterteilung oder Verschmelzung von Körpern kann jedoch nicht davon ausgegangen werden, dass ein dreidimensionales Gitter mit gleichmäßigen Abständen der einzelnen Datenpunkte ausreicht, die Füllstruktur vollständig zu repräsentieren, da dessen Auflösung an verschiedenen Stellen unterschiedlich hoch sein müsste. Dagegen stellt die Anpassung über Ersetzung des Grundkörpers bzw. Veränderung der Wandstärke kein Problem dar, weil Anzahl und Größe der Körper nicht verändert werden. Ein gleichmäßiger Abstand zwischen den Körpern, der essentiell für die Verwendung eines solchen Gitters ist, ist also in diesen Fällen gegeben.

Wie der weitere Verlauf der Arbeit zeigen wird, ist die Umsetzung der Anpassung durch Unterteilung der Grundkörper in diesem Rahmen die geeignete Lösung. Daher muss die zu entwickelnde Datenstruktur eine solche Füllstruktur abbilden können. Die Untersuchung geeigneter Strukturen berücksichtigt dies, wobei eine Erweiterung um andere Anpassungsmethoden jedoch nicht komplett außer Acht gelassen wird. Solche werden an diesem Punkt zweitrangig betrachtet.

Zunächst werden die Zusammenhänge in der Füllstruktur analysiert. Anhand der daraus abgeleiteten Kriterien wird dann eine Eignungsprüfung verschiedener Datenstrukturen vorgenommen.

Nicht-hierarchische Beziehungen zwischen benachbarten Körpern

Jeder Körper der Füllstruktur steht mit mehreren anderen Körpern in einer räumlichen Beziehung. Körper können direkte wie auch indirekte Nachbarn besitzen, wobei sich die direkten Flächen teilen und die indirekten Kanten und Eckpunkte (siehe Abb. 3.13 für eine zweidimensionale Darstellung). Im einfachsten Fall besteht die Füllstruktur aus gleichen Körpern, sodass die Anzahl und Anordnung der benachbarten Körper über die gesamte Füllstruktur (bis auf an den Rändern) ebenfalls gleich ist.

Wird jedoch ein Körper unterteilt, verändern sich die Nachbarschaftsbeziehungen (siehe Abb. 3.12). So gewinnt ein Körper durch die Halbierung eines Nachbarn dann einen weiteren hinzu, wenn die Teilungsebene durch die Kontaktfläche zwischen beiden Körpern verläuft. Wenn die Teilungsebene jedoch parallel dazu verläuft, verändert sich die Anzahl von Nachbarn nicht.

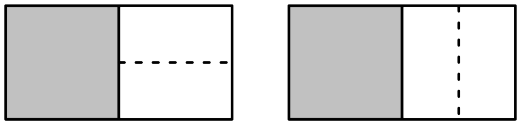


Abbildung 3.12.: Grauer Körper gewinnt Nachbarkörper durch Teilung dazu (links), Anzahl der Nachbarn vom grauen Körper bleibt gleich (rechts)

Durch einen großen Größenunterschied zwischen Körpern entsteht zudem potentiell eine wachsende Ungleichverteilung der Nachbarschaftsbeziehungen. Ein Körper, der aus vielen Teilungen entstammt, also kleiner ist als weniger stark geteilte Körper, besitzt in Nachbarschaft zu einem wenig geteilten, großen Körper, weniger Nachbarn als dieser, wie Abb. 3.14 zeigt. Die Füllstruktur kann demnach lokal stark voneinander abweichende Anordnungen von verschieden großen Körpern aufweisen. Eine Herausforderung an die Datenstruktur, die Beziehungen solcher Art abbilden soll, ist diese potentielle Ungleichverteilung von Daten.

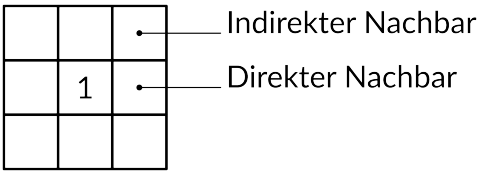


Abbildung 3.13.: Nachbarschaftsbeziehungen von Körper (1) zu gleich großen Körpern

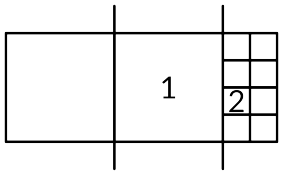


Abbildung 3.14.: Großer Körper (1) ist reich an Nachbarn auf der rechten Seite, während die linke Nachbarschaft von kleinem Körper (2) durch (1) dominiert wird

Hierarchische Beziehungen zwischen Körpern

Genau so wie Körper laterale Beziehungen zu anderen Körpern durch ihre räumliche Nachbarschaft besitzen können, können sie auch über hierarchische Beziehungen verfügen. Ein unmodifizierter Grundkörper ist in dieser Hinsicht ohne Beziehungen. Sobald er verändert

wird, ist er der Parent der von ihm abstammenden Körper, welche als Child dessen Platz einnehmen. Geschieht die Anpassung durch Ersetzen mit einem spezialisierten Körper oder der Veränderung der Wandstärke, geht aus einem Parent nur ein Child hervor (Abb. 3.15). Wenn jedoch eine Teilung des Körpers vollzogen wird, besitzt der ursprüngliche Körper Beziehungen zu mehreren Children, während diese nur einen Parent besitzen (Abb. 3.16. Im Fall der Verschmelzung von mehreren Körpern (Abb. 3.17) verkehrt sich die vorherige Beziehungsanordnung, indem mehrere Parents ein Child erzeugen. Dies in einer Datenstruktur adequat darzustellen, ist nur dann trivial, wenn die Grundkörper die kleinsten Körper darstellen und ein Baum somit rückwärts von den Endknoten zur Wurzel hin aufgebaut werden kann. Bei einer kombinierten Modifikation, worin Körper sowohl geteilt als auch verschmolzen werden können, können sich komplexe Verflechtungen in einem Baum ergeben (siehe Abb. 3.18), zumal auch Körper, die verschiedenen Ästen angehören, kombiniert werden können. Da Körper auseinander hervorgehen, ist ein direkter Verweis auf die obere Hierarchieebene und umgekehrt auch vom Parent auf die Children möglich, wenn ein Körper modifiziert wird.

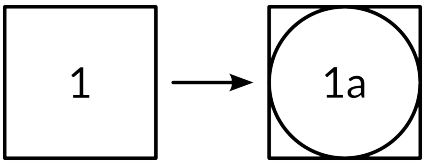


Abbildung 3.15.: Hierarchische Beziehungen zwischen Grundkörper und daraus hervorgehendem spezialisierten Körper

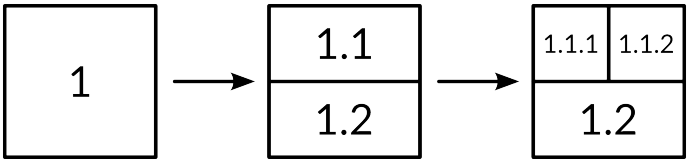


Abbildung 3.16.: Hierarchische Beziehungen zwischen Körpern durch Teilung in Unterkörper

Globale Beziehung zwischen Körper und Füllstruktur

Zuletzt steht ein Körper mit der gesamten Füllstruktur in einer Beziehung. Die Füllstruktur setzt sich aus einer Anzahl von Körpern zusammen, bzw. anders herum betrachtet ist ein Körper ein Teil der Füllstruktur. In dieser besetzt ein Körper eine Position, ordnet sich also

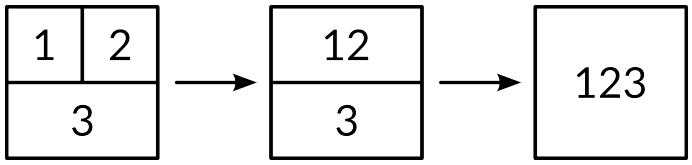


Abbildung 3.17.: Hierarchische Beziehungen zwischen Körpern durch Verschmelzung

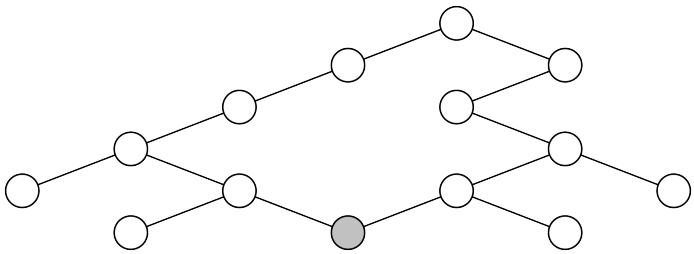


Abbildung 3.18.: Beispielhafter Baum für eine Füllstruktur, die über Teilung und Verschmelzung modifiziert wird. Der graue Knoten geht aus beiden Modifikation hervor und verbindet Äste des Baums.

im globalen Koordinatensystem ein. Darüber hinaus soll die Füllstruktur Aufgaben, die von den Optimierungszielen vorgegeben werden, erfüllen. Da dies lokal über eine Veränderung der Körper erreicht wird, ist die Beziehung zudem eine hierarchische, indem die Ausprägung der Körper durch die zu erfüllende Aufgabe gesteuert wird.

Zugriff und Modifikation

Neben den Beziehungen der Körper muss ebenso die Zugriffsart auf einzelne Körper bedacht werden. Sollen Körper beispielsweise ungeordnet nacheinander abgerufen werden, so sind die Überlegungen zu den Beziehungen unter Umständen hinfällig. Ist beim Zugriff der hierarchische oder lokale Kontext eines Körpers wichtig, müssen diese jedoch bedacht werden.

Durch eine Modifikation der Füllstruktur ändern sich zudem die Beziehungen zwischen einzelnen Körpern. Neue Daten werden erzeugt und müssen in die vorhandene Datenstruktur sinnvoll integriert werden. Eine Datenstruktur, die dieses nicht erlaubt, kann also ohne weitere Betrachtung ausgeschlossen werden.

Zusammenfassung der Kriterien

Die oben dargestellten Zusammenhänge werden in Tabelle 3.1.7 als Optionsmatrix zusammengefasst. Anhand dieser kann abgelesen werden, welche Kontextkombinationen bei der Wahl einer geeigneten Datenstruktur beachtet werden müssen.

Tabelle 3.1.: Darstellungsoptionen für die Datenstruktur

Zugriffskriterien	Kontext	
	global	lokal
Hierarchie	x	x
Nachbarschaft		x
ungeordnet		

3.1.8. Mögliche Datenstrukturen

Je nach Gewichtung der verschiedenartigen Kriterien eignen sich grundsätzlich unterschiedliche Datenstrukturen, um die jeweiligen Beziehungen abzubilden. Diese werden im Folgenden vorgestellt. Vor- und Nachteile sowie die Tauglichkeit für den Einsatzzweck der Strukturen werden abgewogen.

Baumstrukturen

Eine hierarchische Darstellung der Füllstruktur kann über Suchbäume (Knebl, 2019, S. 133 ff.), welche einen Raum in mehrere Unterräume unterteilen, geschehen. Um die Füllstruktur mit einem solchen darzustellen, muss die Hierarchie der Körper auf die einzelnen Knoten des Baums abgebildet werden.

Zunächst können Balancierte Bäume (alternativ Ausgeglichene Bäume genannt, darunter beispielsweise B- und R-Bäume) (Knebl, 2019, S. 140 ff.) ausgeschlossen werden, da diese über den gesamten Baum hinweg eine maximale Tiefe garantieren. Dafür werden Umsortierungsoperationen vorgenommen, welche einzelne Einträge verschieben. Diese Eigenschaft ist für Suchfunktionen vorteilhaft, da dadurch eine begrenzte Laufzeit erreicht wird, aber für die Darstellung der Füllstruktur nicht geeignet, weil die hierarchischen Beziehungen zwischen einzelnen Körpern durch die Umsortierung verloren gehen. Zudem kann eine homogene Verteilung der Verzweigungstiefe nicht die Aufgabe erfüllen, eine lokal anpassbare Füllstruktur mit stark differenzierten Eigenschaften zu erzeugen.

Für eine hierarchische Datenstruktur geeignete Bäume sind der k-d-Baum - ein generischer Suchbaum von k Dimensionen (Abb. 3.19) und der Octree (z.B. von Meagher (1982) beschrieben) (Abb. 3.20). Da diese prinzipiell unbalanciert sind, können beliebige Hierarchien und Hierarchietiefen mit ihnen dargestellt werden. Wu u. a. (2016) haben bereits die Verwendung einer adaptiven räumlichen Anpassung "ähnlich eines Octrees" zur Erzeugung einer Füllstruktur in ihrer Arbeit demonstriert[5] Eine hohe Granularität bei der Volumenänderung durch Viertelung einer Zelle konnte zudem erreicht werden.

Für diese Arbeit kommt jedoch der Octree nicht in Frage, da bei jeder Verzweigung acht Knoten erzeugt werden (entgegen der vier Unterzellen bei Wu u. a. (2016) und Wu (2018)). Es muss bedacht werden, dass in dieser Arbeit die Zellen bzw. Körper auch in der dritten Dimension unterteilt werden können, weshalb eine Halbierung des Körpers eine höhere Granularität gewährleistet. Der binäre k-d-Baum erfüllt diese Anforderung. Wie Abb. 3.19 und 3.20 zeigen ist die Granularität eines binären Baumes im Falle einer regelmäßigen Unterteilung wesentlich geringer als beim Octree. Erst eine sechsfache Unterteilung (als Zahlen in Teilungsergebnis notiert) ergibt den grauen Kubus im k-d-Baum, während ein Octree schon nach zwei Schritten diese Unterteilung erreicht hat. Der Baum ist dadurch flacher, was für den Einsatz in einer Suchfunktion ein schnelleres Erreichen des Ergebnisses, aber auch geringere Flexibilität in der Größe der Unterräume (bei einer regelmäßigen Unterteilung) bedeutet.

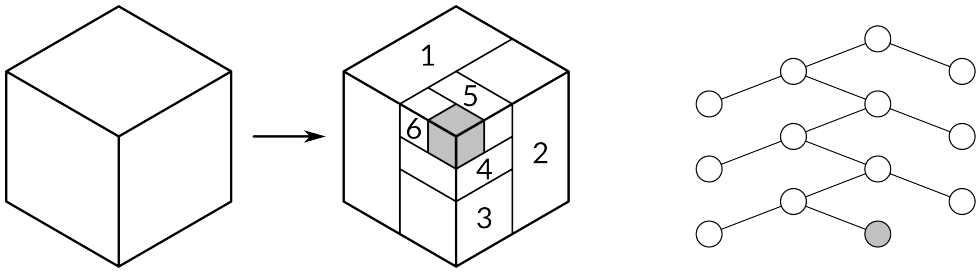


Abbildung 3.19.: Unterteilung eines Raumes mit Hilfe eines k-d-Baumes. Unterräume müssen nicht gleichmäßig aufgeteilt sein.

Knoten der Bäume bezeichnen Grenzen der Unterräume. Zwar unterteilt eine Parkettierung den Raum in änlicher Weise, doch die räumliche Zuordnung der Körper erfolgt nicht über deren Begrenzungen, sondern wie in Abschnitt 3.1.4 dargestellt über einen Referenzpunkt. Die Definition der Baumstruktur muss also dahingehend verändert werden, dass Knoten nicht mehr Grenzen, sondern Körper repräsentieren. Weiterhin wird bei einer Modifikation eines

[5] Da sich die Anpassung auf räumlich extrudierten 2D-Strukturen beschränkt, ist vielmehr der Vergleich zu einem Quadtree (Ottmann und Widmayer, 2017, S. 285 ff.) angemessen, wie die spätere Arbeit von Wu (2018) zeigt.

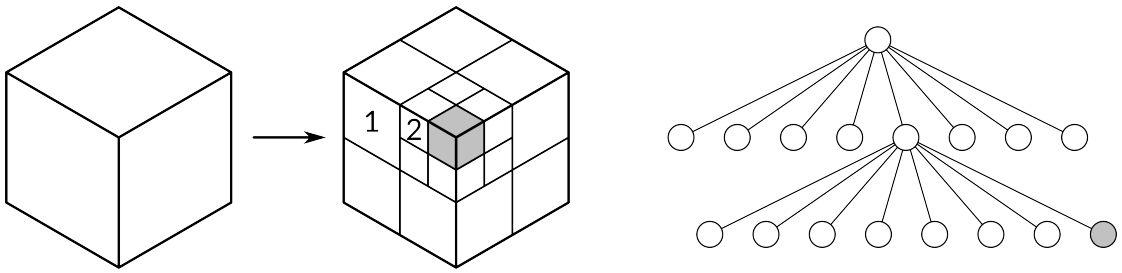


Abbildung 3.20.: Unterteilung eines Raumes mit Hilfe eines Octrees. Unterräume müssen nicht gleichmäßig aufgeteilt sein.

Körpers dieser gelöscht und durch Derivate ersetzt. Der ihm zugeordnete Knoten stellt also zwangsläufig keinen Körper mehr dar, sondern einen Zwischenschritt in der Optimierung. In einer Baumstruktur stellen folglich nur Endknoten Körper dar.

Verkettete Listen

Wird die Füllstruktur über eine verkettete Liste (Dietzfelbinger u. a., 2014, S. 76) repräsentiert, treten die Beziehungen zwischen den einzelnen Körpern in den Vordergrund. Körper werden hier durch Elemente der Liste repräsentiert. Wie bei den Bäumen existieren auch für Listen verschiedene Modelle. Im einfachsten Fall ist eine Liste linear und einfach verkettet (Dietzfelbinger u. a., 2014, S. 82 ff.). Eine vollständige Iteration durch diese erreicht alle gespeicherten Daten, wie Abb. 3.21 zeigt. Das Einfügen von Daten, dargestellt in Abb. 3.22, in eine beliebige Position der Liste ist durch Setzen von Verweisen möglich.

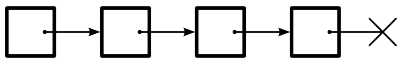


Abbildung 3.21.: Verkettete Liste, letzter Verweis ist ein Null-Pointer

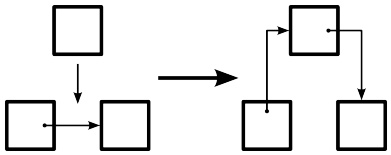


Abbildung 3.22.: Einfügen von neuem Listenelement

Für die Darstellung des lokalen Kontextes in der Füllstruktur ist diese Art von Liste nicht ausreichend, da sie eindimensional ist. Wenn die Beziehungen eines jeden Körpers repräsentiert werden sollen, muss die Liste demnach Verweise zu mehreren Elementen erlauben. Dadurch ergibt sich der Fall der doppelten Verkettung zwischen zwei benachbarten Elementen, da von beiden ein Verweis zum jeweils anderen Element ausgeht. Es handelt sich hier also um einen speziellen, erweiterten Fall einer linearen, doppelt verketteten Liste (Dietzfelbinger u. a., 2014, S. 77 ff.).

Es muss sichergestellt werden, dass durch alle Elemente einer solchen Datenstruktur iteriert werden kann. Im Grundzustand der Füllstruktur ist zwar noch gegeben, dass jedes Element eine bestimmte Anzahl an eindeutig angeordneten Nachbarn hat, doch bereits ab dem ersten Optimierungsschritt gilt dies nicht mehr. Eine Zuweisung von Verweisen basierend nur auf der Richtung des jeweiligen Nachbarn ist also nicht möglich, wie Abb. 3.23 für eine Nachbarschaft mit verschiedenen Körpern zeigt. Zudem müssen bei jeder Modifikation der Füllstruktur die Verweise zwischen einzelnen Elementen aktualisisert werden, was einen erheblichen Rechenaufwand mit sich zieht, da für jedes neu entstandene Element die Nachbarschaftsbeziehungen neu berechnet werden müssen.

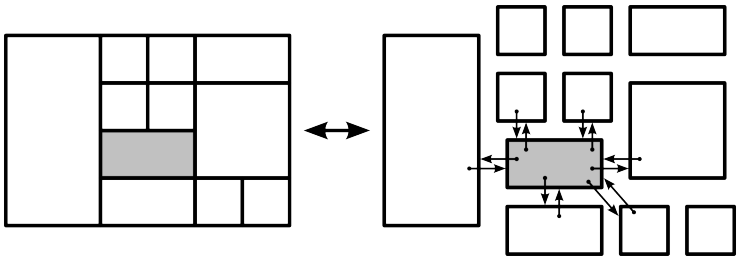


Abbildung 3.23.: Komplexe, zweifach verkettete Liste für zweidimensionale Nachbarschaft von grauem Element nach Modifikation der Füllstruktur

Arrays

Der Einsatz eines n-dimensionalen Arrays zur Repräsenation der Füllstruktur stellt die technisch einfachste Lösung dar. Hierbei liegen Körper als Array-Elemente vor. Wenn die Füllstruktur durch einen linearen, eindimensionalen Array repräsentiert wird, entspricht die Datenstruktur nicht der Ausprägung der Füllstruktur. Dafür kann sehr einfach durch diese iteriert werden. Die meisten Eigenschaften einer einfach verketteten, linearen Liste (siehe oben) treffen auch auf diese Art von Array zu. Das Einfügen von Daten zwischen Array-Einträgen ist jedoch ohne Verschiebung des nachfolgenden Array-Teils nicht möglich (siehe Abb. 3.24). Eine solche Lösung kann beispielsweise dann zum Einsatz kommen, wenn nicht gefordert

ist, dass das Framework Kenntnisse über die Anordnung der Körper besitzen muss und diese durch andere Methoden eindeutig identifiziert werden können.

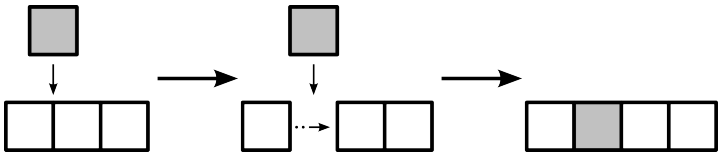


Abbildung 3.24.: Einfügen von neuem Array-Element

Die Repräsentation durch ein dreidimensionales Array gibt Aufschluss über die Nachbarschaftsbeziehungen der Körper im Ursprungszustand, da diese als Array-Elemente mit aufeinander folgenden Indizes vorliegen. Wie schon in Abschnitt 3.1.7 dargestellt, kann dieses Datenmodell die Füllstruktur nicht mehr adequat darstellen, wenn die Optimierung dazu führt, dass Körper geteilt oder miteinander kombiniert werden. Es müsste eine konstante Umsortierung im Array geschehen, wenn der Platz im Array mit der örtlichen Anordnung des Körpers zusammenfallen soll.

Eine Kombination aus einem dreidimensionalen Array und Bäumen kann jedoch die oben genannten Probleme für den Fall der Modifikation durch Teilung lösen: Die Grundkörper werden durch Array-Elemente repräsentiert. Sobald sie geteilt werden, nimmt ein Baum, der sich mit jeder Teilung weiter verzweigt, deren Platz ein, wie in Abb. 3.25 dargestellt. Dadurch kann sowohl eine hierarchische als auch eine regionale (wenn auch keine konkret nachbarschaftliche) Beziehung eines Körpers dargestellt werden, ohne das Array in seiner Größe zu verändern.

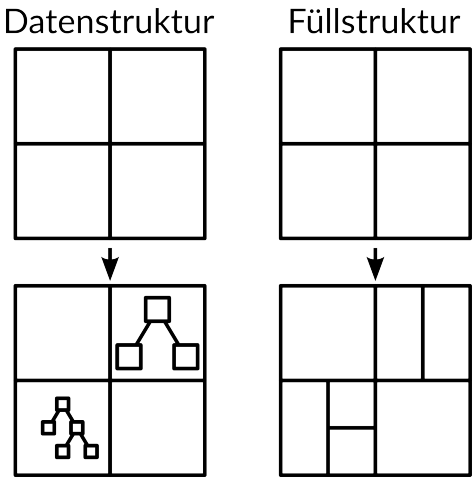


Abbildung 3.25.: Kombination eines n-dimensionalen Arrays mit binären Bäumen

Vergleich der Datenstrukturen

Ist beispielsweise gewünscht, dass bei der Optimierung der unmittelbare lokale Kontext berücksichtigt wird, eignet sich vor allem eine mehrfach verkettete Liste. Soll die Optimierungshistorie erhalten bleiben, ist der Einsatz von Bäumen unumgänglich. Den geringsten Programmieraufwand stellt dagegen die Implementierung mit einem linearen Array dar. Falls keine weiteren Features benötigt werden, kann dies also ebenfalls eine legitime Lösung sein.

Tabelle 3.1.8 liefert einen Überblick über die möglichen Lösungen. Dabei sind die in Spalten geordneten Kriterien nicht als gleichwertig zu betrachten, da nicht jedes Kriterium auf jede Gesamtlösung angewandt werden kann. Je nach Implementierung kann ein Kriterium also das ausschlaggebende sein.

Tabelle 3.2.: Vergleich der vorgestellten Datenstrukturen

	Programmieraufwand	Nachbarschaft	Historie
Baum	-	0	++
Einfach Verkettete Lineare Liste	+	--	--
Mehrfach Verkettete Liste	--	++	--
Eindimensionales Array	++	--	--
Dreidimensionales Array	++	0	--
Dreidimensionales Array + Baum	-	+	++

3.2. Benötigte Daten

Aufgrund der oben beschriebenen Anforderungen an die Füllstruktur kann eine Analyse der benötigten Daten getätigt werden. Diese teilen sich in folgende grobe Kategorien auf:

- Grundeinstellungen des Frameworks bzw. der Füllstruktur
- lokale Eigenschaften / Optimierungsziele
- vom Programmierer einzugebene Befehle und Informationen

Die Kategorien sind so gewählt, dass sie verschiedene Schritte des Programmablaufs und die Spezifizierung von Informationen repräsentieren. Während die Grundeinstellungen allgemeine Informationen enthalten und das Framework für die Optimierung nach Benutzervorgaben vorbereiten, präzisieren die Optimierungsziele die Vorgaben und differenzieren sie örtlich. Zuletzt kann der Programmierer die mit den vorherigen Schritten gelieferte Informationen über Befehle konkret in Aktionen des Frameworks umsetzen.

3.2.1. Grundeinstellungen

Um das Framework zu initialisieren, werden Grundeinstellungen benötigt. Diese müssen Informationen über gewünschte anfängliche Ausprägung der Füllstruktur liefern, wie auch Einschränkungen vorschreiben, über die die Optimierung nicht hinausgehen sollte. Die Grundeinstellungen bestehen somit aus mehreren Datensätzen, die jeweils eine kleine Anzahl an Daten beinhalten. Geeignete Datenformate zur Darstellung der Grundeinstellungen werden im Abschnitt 3.3 vorgestellt.

Benötigter Optimierungsraum

Das Framework soll mit weiteren, in dieser Arbeit nicht umgesetzten Programmteilen Anweisungen für 3D-Drucker generieren. Die Limitationen der eingesetzten Hardware werden daher benötigt, um den Optimierungsraum bzw. die Ausprägung der Füllstruktur einzuschränken. Die räumliche Begrenzung des zu optimierenden 3D-Modells kann in einem der Optimierung vorgeschalteten Schritt erfolgen, sodass das Framework nur den benötigten Optimierungsraum als Begrenzungskörper (in der Regel ein Quader) des 3D-Modells erhalten muss.

Extrema der Ausprägung einzelner Körper

Extrema der Ausprägung einzelner Körper der Füllstruktur sind vom jeweiligen 3D-Modell wie auch dem 3D-Drucker abhängig. Zusätzlich ist für jede Methode der Modifikation von Körpern eine extreme Ausprägung anders definiert. Körpergröße wie auch Wandstärke können numerisch beschrieben werden. Bei einer Modifikation durch Ersetzen des Körpers mit spezialisierten Körpern muss die Ausprägung jedoch auf eine andere Art mit Grenzen verknüpft werden. Dem Framework müssen bei der Initialisierung und Optimierung diese Parameter bekannt sein, damit es den Benutzereinstellungen entsprechende Ergebnisse liefern kann.

Verwendete Hardware

Auf gleiche Weise können andere Parameter wie minimale/maximale Wandstärke etc. durch die verwendete Hardware beeinflusst werden. Die Füllstruktur muss an verschiedene Verfahren angepasst werden können (siehe auch Abschnitt 3.1). Da im Rahmen dieser Arbeit nur das FFF-Verfahren betrachtet wird, werden die Hardware-Daten zunächst ignoriert.

Werkstoffdaten

Um eine Optimierung der Füllstruktur vorzunehmen, müssen einige Werkstoffdaten bekannt sein. Mit diesen können physikalische Eigenschaften eines Körpers berechnet werden, um sie einem Optimierungsziel gegenüberzustellen. Daher sollte der benötigte Optimierungsraum sowie die Extrema der Ausprägung einzelner Körper den Grundeinstellungen zugeordnet werden.

Weiterhin können Werkstoffdaten vom aufrufenden Programmteil definiert werden, wenn diese über die Grundeinstellungen übermittelt werden, während der Programmierer der Optimierungsvorschriften verschiedene Werkstoffe und ihre Eigenschaften in seinen Vorschriften implementieren muss, wenn sie mit diesen gebündelt würden. Es muss zudem bedacht werden, dass im Falle eines aus mehreren Werkstoffen bestehenden Modells eine Möglichkeit existieren sollte, diesen Umstand beim Einsatz des Frameworks zu berücksichtigen. Zwar wird dieser Sonderfall im Rahmen dieser Arbeit nicht weiter betrachtet, doch eine Erweiterung des Frameworks, um ihn abzudecken, sollte in Zukunft möglich sein.

3.2.2. Optimierungsdatensatz

Gewünschte Eigenschaften der Füllstruktur bzw. Optimierungsziele und/oder anliegende Faktoren wie mechanische Belastungen sollen in einem Datensatz, hier Optimierungsdatensatz genannt, dargestellt werden, um eine Optimierung der Füllstruktur zu ermöglichen. Diese Eigenschaften und Ziele müssen zwecks der lokalen Optimierung in einer Weise räumlich geordnet vorliegen.

Struktur des Optimierungsdatensatzes

In der Regel eignet sich ein räumliches Gitter gut für die Repräsentation eines Datensatzes, der räumlich verteilte Eigenschaften darstellen soll. Dieses Gitter kann regelmäßig angeordnet sein oder dort, wo eine höhere Auflösung benötigt wird, eine höhere Dichte aufweisen (siehe Abb. 3.26). Der Vorteil des ersten Ansatzes ist, dass bei einer Iteration über den Datensatz gleichzeitig auch die räumliche Position eines Gitter-Elementes, im Folgenden Datenpunkt genannt, inferiert werden kann, wenn dessen Position im Gitter bekannt ist.

Wird im Datensatz eine hohe Auflösung nur an einigen Stellen gefordert, muss dafür das komplette Gitter diese hohe Auflösung besitzen, was zu einem höheren Berechnungsaufwand und Speicherbedarf führt. Dagegen muss ein Gitter mit variabler Dichte immer Informationen über die räumliche Position eines Datenpunktes mitliefern, da dessen Position im Gitter keine Aussagen mehr darüber zulässt. Die eigentlichen Optimierungsziele sind unbe-

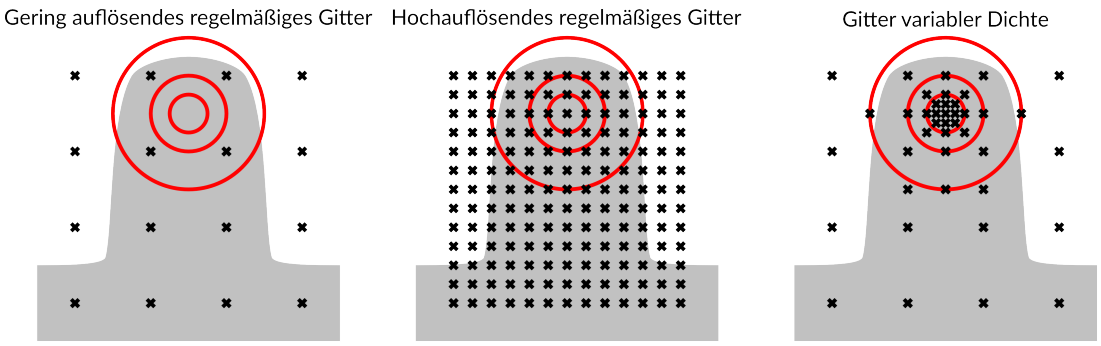


Abbildung 3.26.: In Gittern angeordnete Datenpunkte (schwarz). Die Gitter sind einem Modell (grau) zugeordnet, dessen besondere Regions of Interest (rot) einer hohen Auflösung bei der Optimierung bedürfen.

kannter Anzahl und Ausprägung und werden durch die Anwendung des Frameworks gegeben. Daher muss ein Datenpunkt ähnlich wie ein Körper der Füllstruktur so gestaltet sein, dass sie diese Flexibilität in der Umsetzung ermöglichen. Beispielsweise können gespeicherte Daten die gewünschte lokale Dichte sein oder auch mechanische Belastungen (der Einfachheit halber wird hier nur von Optimierungszielen gesprochen, obwohl mechanische Belastungen an sich kein Ziel, sondern der Anlass zur Optimierung sind). Die Dichte kann als Skalar dargestellt werden, während die Belastungen eines Vektors (wenn nur Kräfte betrachtet werden) oder einer Matrix (Kräfte und Torsion) bedürfen. Auf geeignete Datenformate für den Optimierungsdatensatz wird im Abschnitt 3.3 eingegangen.

Inhalte des Optimierungsdatensatzes

Die Datenpunkte des Optimierungsdatensatzes müssen neben Optimierungszielen auch Daten enthalten, anhand derer sie eindeutig identifiziert und abgerufen werden können. Eine einzigartige ID oder das Speichern des Datenpunktes als Schlüssel-Wert-Paar, wobei der Schlüssel als eindeutiger, idealerweise deskriptiver Identifikator fungiert, kann dazu dienen, dies zu erreichen. Wie in oben beschrieben, ist eine räumliche Zuordnung des Datenpunktes unter Umständen nötig, weshalb die Position des Datenpunktes im Referenzsystem gespeichert werden können soll.

Einzelne Datenpunkte müssen alle nötigen Daten enthalten, anhand derer die Optimierung der Füllstruktur gesteuert wird. Diese Daten, auch Optimierungsziele genannt, sind in der Regel mathematische und physikalische Größen wie beispielsweise die maximale Größe oder die Dichte eines Körpers, Lichtdurchlässigkeit, mechanische Belastungen usw. Wie im

Abschnitt 3.2 erwähnt wird, muss das 3D-Modell dem Framework nicht bereitgestellt werden, wenn aus dem Optimierungsdatensatz nötige Informationen hervorgehen. Die Distanz eines Datenpunktes zur Modellaußenhülle kann in diesem Fall genutzt werden, diese zu approximieren. Bei einer genügend hohen Auflösung des Optimierungsdatensatzes kann die Optimierung der Füllstruktur dann aufgrund der in den Datenpunkten gespeicherten Distanz erfolgen. Da das Framework die Anwendung nicht vorgeben soll, wird neben diesen Beispielen an dieser Stelle nicht weiter auf konkrete Optimierungsziele eingegangen.

Wie die Grundeinstellungen besteht der Optimierungsdatensatz aus mehreren einzelnen Datensätzen, den Datenpunkten, die ebenfalls jeweils eine geringe Menge an Daten beinhalten. Strukturell sind beide Datensätze also recht ähnlich, sodass die Möglichkeit besteht, ein einzelnes Datenformat auszuwählen, welches verwendet weren kann, um diese darzustellen. Die weitere Auslegung eines geeigneten Datenformates findet in Abschnit 3.3 statt.

3.2.3. Optimierungsvorschriften

Die Optimierungsziele werden erst durch den Optimierungsdatensatz dem Framework bekannt. Wie in Abschnitt 3.2.2 angedeutet, sind die Optimierungsziele stark unterschiedlicher Natur, weshalb das Framework die Optimierung nicht über eine für alle Ziele anwendbare Funktion durchführen kann. Beispielsweise kann die durchschnittliche Dichte eines Körpers mit der vorgegebenen verglichen werden, doch um die Reaktion eines Körpers auf mechanische Beanspruchung zu ermitteln, bedarf es komplexerer Berechnungen. Deshalb muss der Programmierer die Funktionalität liefern, mithilfe derer die Optimierung berechnet werden kann. Für jedes Optimierungsziel wird also eine Berechnungsvorschrift, im Folgenden Optimierungsvorschrift genannt, benötigt, die Daten von der Füllstruktur und dem Optimierungsdatensatz verarbeiten können muss.

Die Optimierungsvorschriften sollen so in das Framework integriert werden können, dass, wenn alle benötigten Daten vorhanden sind, neben der Einbindung kein weiterer Programmieraufwand nötig sein soll, den Datenfluss zwischen Framework und Vorschrift zu ermöglichen. Um also die entsprechenden Daten aus Füllstruktur und Optimierungsdatensatz vom Framework zu erhalten, muss eine Optimierungsvorschrift Instruktionen liefern, mit Hilfe derer sie Daten auswählen oder anfragen kann.

Da das Framework eine Entscheidung zur weiteren Optimierung oder zum Anhalten dieser benötigt, müssen Optimierungsvorschriften Rückgabewerte liefern, die aus der Verarbeitung der eingespeisten Daten entstammen und diese Entscheidung steuern können. Ferner müssen Optimierungsvorschriften und deren Rückgabewerte in einem Format vorliegen, dass das Framework eine beliebige Vorschrift ohne Abänderung ihrer eigenen Programmstruktur verwenden kann. Für Optimierungsvorschriften muss also eine Schnittstelle geschaffen werden.

3.2.4. Hilfsfunktionen

Weiterhin werden Hilfsfunktionen benötigt, mit denen Zusammenhänge berechnet werden können, die nicht unmittelbar aus den der Füllstruktur oder dem Optimierungsdatensatz vorgegebenen Daten hervorgehen. Ein Beispiel hierfür ist die durchschnittliche Dichte eines Körpers, zusammengesetzt aus der Dichte des Wandmaterials und der Luft im Hohlraum. Durch die Modifikation eines Körpers verändern sich die Massenanteile von Wandmaterial und Hohlraum, sodass die durchschnittliche Dichte immer wieder berechnet werden muss, wenn durch die Optimierung der Füllstruktur der Körper verändert wird. Da solche Zusammenhänge unter Umständen an mehreren Stellen benötigt werden, werden Hilfsfunktionen benötigt, die solche Berechnungen durchführen und an entsprechenden Stellen im Programmablauf geladen werden können.

Ähnlich wie die Optimierungsvorschriften benötigen die Hilfsfunktionen Daten aus der Füllstruktur und/oder dem Optimierungsdatensatz und liefern einen Rückgabewert, der jedoch in diesem Fall weniger stark auf ein bestimmtes Format eingeschränkt werden muss. Ebenso soll die Integration einer Hilfsfunktion mit möglichst geringerem weiteren Aufwand verbunden sein, sodass ein ähnlicher struktureller Aufbau wie bei den Optimierungsvorschriften denkbar ist.

3.2.5. 3D-Modell

Das 3D-Modell, für welches die Füllstruktur erzeugt werden soll, hat auch bei traditionellen Slicern Einfluss auf die Ausprägung der Füllstruktur. Im einfachsten Fall bedeutet dies die Beschneidung der Füllstruktur auf die Maße des Modells, doch Slicer können beispielsweise bereits erkennen, wenn ein Modellquerschnitt zu klein ist, als dass eine Füllstruktur sinnvoll wäre und diesen komplett mit Material ausfüllen[6]. Abschnitt 3.2.2 legt dar, dass für einige Anwendungen durchaus eine Alternative zum Laden des 3D-Modells in das Framework besteht. Wird jedoch Informationen aus dem 3D-Modell selber, beispielsweise der Winkel einer Facette der Modelloberfläche zum Druckbett, für eine Optimierung benötigt, ist die Übermittlung des Modell unter Umständen unumgänglich.

Für das Slicing und den Beschnitt der Füllstruktur auf das 3D-Modell wird dieses jedoch nicht vom Framework benötigt, da jenes nur auf die Erstellung einer Füllstruktur ausgelegt ist. Durch das Ausschließen des 3D-Modells von der Verarbeitung durch das Framework würde dessen Implementierung vereinfacht. Die Beschränkung der Schnittstellen reduziert

[6] https://manual.slic3r.org/expert-mode/print-settings, Abschnitt Ïnfill Optimization: Solid infill threshold area"

ebenfalls den Aufwand beim Einsatz des Frameworks. Andererseits könnten durch das Einschränken der Schnittstellen Anwendungsgebiete und Anpassungsmöglichkeiten beschnitten werden, was dem Ziel einer hohen Flexibilität des Framework entgegensteht.

3.3. Datenformate

Wie aus den Anforderungen in Abschnitten 3.2.1 und 3.2.2 hervorgeht, wird ein flexibles Datenformat benötigt, das es ermöglicht, verschiedenartige Daten unbekannter Anzahl und Art zu repräsentieren. Diese Daten weisen zudem eine Hierarchie auf und sind unter Umständen selber Datensätze.

Bei der Wahl des Datenformates fällt die erste Entscheidung zwischen proprietärem und offenen Standard. Beide Wege haben ihre eigenen Vor- und Nachteile. Einerseits kann ein proprietäres Format genau auf die eigenen Anforderungen zugeschnitten werden und somit effizienter die gewünschten Daten speichern. Andererseits würde dieses einen höheren Entwicklungsaufwand bedeuten, da das Datenformat definiert werden muss und Schnittstellen und Softwarebibliotheken dafür geschrieben werden müssen. Zusätzlich muss sichergestellt werden, dass das Format die Daten fehlerfrei speichern kann.

Ein offener Standard ist dagegen nicht auf die Aufgabe exakt zugeschnitten, spart aber Entwicklungsaufwand ein. Ein Programmierer, der das Framework benutzt, kann außerdem oftmals auf bereits existierende Softwarebibliotheken zurückgreifen, sodass der Entwicklungsaufwand auch für ihn reduziert wird. Im Rahmen dieser Arbeit ist es also durchaus sinnvoll, auf einen offenen Standard zurückzugreifen. Dies senkt zudem die Hemmschwelle für andere Entwickler, das Framework in ihren Projekten zu verwenden oder zur Entwicklung des Frameworks beizutragen. Verschiedene Dateiformate sind prinzipiell dafür geeignet, komplexe Datensätze zu serialisieren.

Strukturell am einfachsten ist das von Shafranovich (2005) definierte Comma Separated Values (CSV) Format. In diesem werden Werte in einer festen Struktur durch Trennzeichen und Zeilenumbrüche voneinander separiert. Ein Gruppieren und Verschachteln der Daten ist nicht möglich, daher ist das CSV-Format auf eine zweidimensionale Struktur beschränkt, was dieses Format für den Einsatzzweck unbrauchbar macht.

In der von Crockford (2017) definierten Javascript Object Notation (JSON) formatierte Daten sind menschenlesbar und hierarchisch. Daten liegen als Schlüssel-Wert-Paare vor und können verschachtelt werden. Eine solche Formatierung ist für den Einsatzzweck vorteilhaft, wie in Abschnitt 3.2.2 diskutiert worden ist. Für das JSON-Format existieren in verschiedenen Programmiersprachen Bibliotheken, sodass die Implementierung einer Schnittstelle mit geringem Aufwand geschehen kann.

Als ein binär codiertes Format ist das von Furuhashi (2008) entwickelte MessagePack im Gegensatz zu den anderen hier vorgestellten Formaten nicht menschenlesbar. Der Fokus liegt jedoch auf einer möglichst effizienten und kompakten Datenübertragung durch die Optimierung der eingesetzten Datentypen. Wie bei JSON können komplexe Datenstrukturen abgebildet werden.

Die von Bray u. a. (2013) definierte Extensible Markup Language (XML) ist darauf ausgelegt, erweiterbar zu sein und bildet so die Grundlage für viele Dateiformate mit sehr unterschiedlichen Anwendungshintergründen. Beispielsweise basieren das Vektorgraphikformat SVG[7] wie auch verschiedene Büroanwendungsformate[8] [9] auf XML. Ähnlich wie JSON ist XML menschenlesbar.

Durch die Auslegung als ein menschenlesbares und -editierbares Format für Konfigurationsdateien bietet die von Ben-Kiki u. a. (2009) entwickelte YAML Ain't Markup Language (YAML) ähnliche Eigenschaften wie JSON und XML. Die einfache Struktur des Dokumentes ermgölicht die schnelle Erfassung der Datenstruktur durch Menschen. Interessante Features sind die Fähigkeit, Daten über Label innerhalb der Datei zu referenzieren, wie auch das Speichern von binären Daten und das Deklarieren von eigenen Datentypen.

3.4. Ausgabeformat

Um die Füllstruktur aus dem Framework in weitere Programmteile zu exportieren, wird eine Datenschnittstelle benötigt. Im Rahmen dieser Arbeit wird die Weitergabe der Daten zwar nicht im Detail behandelt, doch um ein Funktionieren des Frameworks nach Vorgabe zu überprüfen und dieses zu debuggen, müssen Ausgaben generiert und untersucht werden können. Daher sollte bereits ein Prototyp einer Ausgabe vorliegen und erste Entwicklungsschritte zu einem Ausgabeformat vollzogen sein.

3.4.1. Ausgabe der Füllstruktur

Die primäre Funktion der Ausgabe ist das Weitergeben der in der Füllstruktur enthaltenen Daten. Dabei muss eine Entscheidung zwischen einer die Füllstruktur wiederspiegelnden Datenstruktur oder Schichtinformationen getroffen werden. Wird erstere Option gewählt, spezialisiert sich das Framework komplett auf die Erzeugung der Füllstruktur. Nachfolgende Programmteile müssen dann das Slicing umsetzen. Ein Vorteil dieses Ansatzes ist, dass durch

[7] https://developer.mozilla.org/en-US/docs/Web/SVG
[8] http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf
[9] docs.oasis-open.org/office/v1.2/OpenDocument-v1.2-part1.pdf

die übertragenen Rohdaten eine hohe Flexibilität gegeben ist. Die Slicing-Software kann dann gezielt Entscheidung bei der Umsetzung der Füllstruktur in Schichten treffen, wie auch beispielsweise die Schichtdicke verändern, ohne dass ein erneuter Export nötig wäre.

Für den direkten Export von Schichtinformationen spricht, dass für Programmierer, die das Framework in ihrem Projekt einsetzen wollen, ein geringerer Aufwand entsteht. Dies kann ebenfalls dadurch erreicht werden, wenn ein Export-Tool mit dem Framework mitgeliefert wird, welches die Schichtinformationen generiert. Daher kann schon hier die Entscheidung für den direkten Export der Füllstruktur als solche getroffen werden.

3.4.2. Exportierte Daten

Die Füllstruktur enthält mehr Daten als für die Erzeugung von Schichtinformationen nötig ist. Wenn diese Daten mit exportiert werden, kann die Füllstruktur rekonstruiert werden. Beispielsweise ist dies sinnvoll, wenn durch sie in ihrer Gänze gespeichert und zu einem anderen Zeitpunkt erneut einer Optimierung unterzogen werden soll. Andererseits beschränkt die Menge an ausgegebenen Daten die Verarbeitungsgeschwindigkeit, sodass die Effizienz bei einer uneingeschränkten Ausgabe geringer ist als bei einer selektiven. Wie auch bei der Wahl zwischen Export der Füllstruktur oder von Schichtinformationen kann durch ein weiteres Tool die Datenmenge nach dem Export eingegrenzt werden, wenn dies nötig sein sollte. Gerade im Prototypenstadium sollte eine möglicht generelle Lösung angestrebt werden, die in folgenden Schritten optimiert wird.

3.4.3. Debugging des Frameworks

Die sekundäre Funktion der Ausgabe liegt im Debugging und Verifizieren der Ausgabe des Frameworks. Da die Füllstruktur eine hohe räumliche Komplexität aufweisen und der dabei entstehende Datensatz an Körpern umfangreich sein kann, ist es ratsam, eine Visualisierung der Ausgabe zu implementieren. Diese lässt sich von Menschen schnell erfassen[10] und kann Aufschlüsse über Fehler in der Optimierung liefern.

Eine Visualisierung kann in 2D oder 3D geschehen. Durch eine zweidimensionale Schnittdarstellung können innere Strukturen offenbart werden. Um dies ebenso zu erreichen, muss eine 3D-Ansicht Begrenzungen der Sichtbarkeit implementieren (siehe Abb. 3.27). Dafür ist es möglich, die Füllstruktur aus mehreren Perspektiven zu betrachten. Während eine Schnittdarstellung mit Hilfe eines einfachen Bildbetrachtungsprogramms aufrufbar ist, sind für die Betrachtung von 3D-Modellen spezielle Programme nötig. Zudem müssen diese wie oben

[10] http://news.mit.edu/2014/in-the-blink-of-an-eye-0116

beschrieben eine Begrenzung der Sichtbarkeit erlauben und die räumliche Orientierung des Modells dem Benutzer kommunizieren, um uneindeutige Ansichten zu vermeiden.

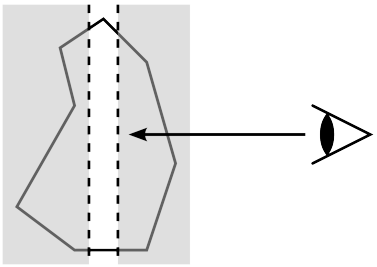


Abbildung 3.27.: Begrenzung der Sichtbarkeit bei einem 3D-Modell. Einsatz von minimaler und maximaler Sichtweite (hier planar) führt zu einer Pseudo-Schnittdarstellung.

Die Füllstruktur ist in diesem Schritt noch nicht mit dem zugehörigen 3D-Modell kombiniert. Für das Debugging muss jedoch bekannt sein, wie die Füllstruktur im 3D-Modell angeordnet ist, weshalb eine Visualisierung der Füllstruktur mit dem entsprechend ausgerichteten 3D-Modell nötig ist.

3.5. Testdateien

Um zu verifizieren, dass das Framework optimierte Füllstrukturen nach Vorgabe erzeugt, müssen Testdateien konzipiert werden. Dazu werden Optimierungsdatensätze benötigt, die eindeutige und mit geringem Aufwand überprüfbare Ergebnisse bei der Optimierung liefern. Die Gestaltung dieser wird durch die zu verifizierenden Kriterien bestimmt, welche im Folgenden aufgelistet sind:

- Erreichen der Optimierungsziele
- Beenden der Optimierung bei Erreichen der Optimierungsziele
- Anwendbarkeit von mehreren lokal differenzierten Optimierungszielen in einer Füllstruktur
- Einhalten von Grenzbedingungen
- Korrekte Ausrichtung der Füllstruktur (keine Transformationsfehler)
- Anwendbarkeit auf Vielzahl von Optimierungsvorgaben (Generalisierbarkeit)

4. Design

Anhand der ermittelten Anforderungen (siehe auch Anforderungsliste im Anhang) wird das Framework entworfen. Dafür werden Datenstrukturen und -Formate für die einzelnen Komponenten ausgewählt sowie ein Algorithmus entwickelt, der die Optimierung der Füllstruktur steuert. Damit die erfolgreiche Optimierung der Füllstruktur verifiziert werden kann, wird ein Testschema konzipiert.

4.1. Füllstruktur

Die diskutierten Möglichkeiten, eine Füllstruktur zu erzeugen, bieten spezifische Vor- und Nachteile. Eine Kombination der verschiedenen Ansätze würde zu einem optimalen Ergebnis führen, da dann nicht nur die Füllstruktur angepasst, sondern ihre Anpassung selber gesteuert werden kann. Im Rahmen dieser Arbeit wird sich auf einen Ansatz konzentriert, um einen Software-Prototypen herzustellen. Die Umsetzung sollte jedoch Möglichkeiten der Erweiterbarkeit offen lassen, sodass in einem weiteren Schritt das Framework ausgebaut werden kann.

4.1.1. Wahl der Anpassungstechniken

Die Auflösung der Füllstruktur bestimmt den Grad der Anpassbarkeit. Wird das anfängliche Grundkörpergitter mit maximalen Abmessungen initialisiert, kann dieses feine lokale Unterschiede in den Optimierungszielen oder Eigenschaften nicht abbilden, wenn Grundkörper durch spezialisierte Körper ersetzt werden oder wenn die Wandstärke verändert wird. Andererseits bedeutet ein feines Grundkörpergitter zwar eine bessere lokale Anpassbarkeit, doch mitunter kann die resultierende Füllstruktur dann in der Dichte schlecht optimiert sein. Eine Parkettierung des Grundkörpers durch mehrere kleinere Körper (bzw. dessen Unterteilung) ist hier die einzige Möglichkeit, eine hohe Auflösung der Anpassbarkeit zu erreichen, ohne dabei grobe Strukturen auszuschließen.

Der Nachteil einer Parkettierung bzw. Unterteilung ist, dass mit steigender Optimierungsstufe tendentiell immer mehr Körper bei der Berechnung der Füllstruktur in Betracht gezogen

werden müssen. Somit ist diese Methode die am meisten rechenintensive. Ein Ersetzen mit vorher berechneten spezialisierten Körpern ist in dieser Hinsicht die optimale Lösung.

Da ein möglichst breites Einsatzspektrum gegeben sein soll, wird die Parkettierung/Unterteilung als Lösungsansatz umgesetzt. In einer weiteren Ausarbeitung kann dann auch beispielsweise ein aus der Unterteilung hervorgegangener Körper dazu dienen, von spezialisierten Körpern ersetzt zu werden. Hier würden die Vorteile der höheren Auflösung mit denen einer spezifischeren Anpassung kombiniert werden können (siehe auch Abschnitt 6.2). Weiterhin kann bei diesem Ansatz zusätzlich die Wandstärke der Körper variiert werden, was ihn zu dem flexibelsten unter den vorgestellten macht.

4.1.2. Wahl des Körpers

Da der Quader von den möglichen geometrischen Körpern den geringsten Berechnungsaufwand benötigt und die geringste Isotropie beim Anfangszustand der Füllstruktur erzeugt, wird er für die Umsetzung als dem der Füllstruktur zugrundeliegenden Körper gewählt. Um einen Körper im Raum zu lokalisieren, muss er mit einem Referenzpunkt versehen werden, wie in Abschnitt 3.1.4 bereits beschrieben worden ist. Prinzipiell eignen sich Eckpunkte oder der Mittelpunkt des Quaders als Referenzpunkt. Darüber hinaus kann jeder beliebiger Punkt als Referenz dienen, doch ein solcher weist in der Regel keine Vorteile gegenüber den zwei vorgestellten Ansätzen auf und ist nicht intuitiv mit besonderen Merkmalen der Körpergeometrie verknüpfbar.

Eckpunkt als Referenz

Wird ein Eckpunkt als Referenz gewählt, erstreckt sich der Körper im lokalen Koordinatensystem nur in eine Richtung. Idealerweise kann dann der Eckpunkt gewählt werden, der zu einer rein positiven Ausdehnung führt. Unter Verwendung der Dimensionen des Körpers kann mit minimalem Aufwand allein durch Einsetzen dieser die Position aller Eckpunkte ermittelt werden.

Mittelpunkt als Referenz

Durch die Wahl des Zentrums als Referenzpunkt muss die Position eines jeden Eckpunktes berechnet werden. Allerdings erleichtert dieser Referenzpunkt die Berechnung von anderen Werten, wie beispielsweise die mittlere Entfernung von zwei verschieden großen Körpern oder die Entfernung zwischen Körper und Außenhülle des 3D-Modells. Bei einer Erweiterung des Frameworks um weitere mögliche Körper hat dieser Ansatz den weiteren Vorteil, dass er

universell einzusetzen ist. Der Mittelpunkt eines jeden Körpers ist immer gleich definiert, wohingegen es fraglich ist, ob der Koordinatenursprung beispielsweise bei einem sechsseitigem Prisma auf einem Eckpunkt liegen sollte oder außerhalb des Prismas auf dem Eckpunkt des Begrenzungsquaders, um die räumliche Ausdehnung in eine Richtung zu beschränken, wie in Abb. 4.1 dargestellt. Daher wird der Mittelpunkt trotz zunächst höheren Rechenaufwandes als Referenz gewählt.

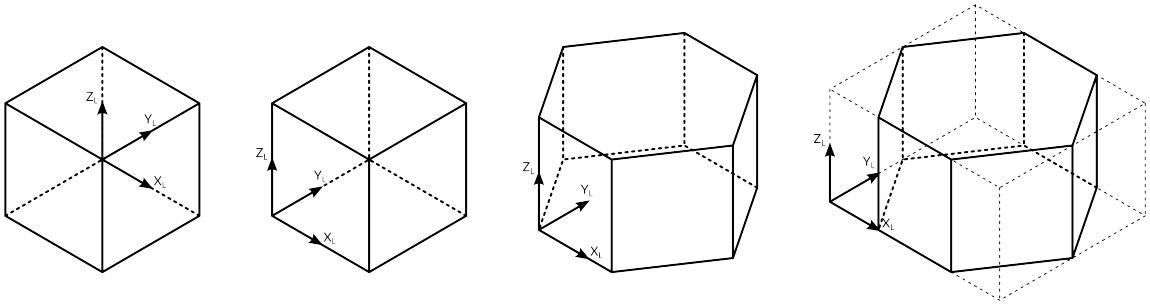


Abbildung 4.1.: Koordinatenursprung mittig im Quader und an Eckpunkten von Quader, hexagonalem Prisma und dessen Begrenzungsquader

4.1.3. Im Körper gespeicherte Daten

Um die aktuelle Füllstruktur mit den Optimierungszielen zu vergleichen, muss diese eine Möglichkeit besitzen, für die Optimierungsziele relevante Daten zu speichern. Neben Position, Dimension und Wandstärke muss das Datenmodell eines Körpers also einen Speicher für eine unbekannte Anzahl an weiteren Werten von unbekanntem Typ implementieren. Die Wandstärke ist dabei deshalb nicht optional, da ihr minimaler Wert vom Durchmesser der Extruderdüse bestimmt wird und somit technisch bedingt ist (vergleiche auch Rezaie u. a. (2013), Wu u. a. (2016) und Wu (2018)).

Die Identifikation der weiteren Werte muss trotz der Tatsache, dass sie vom Benutzer des Frameworks frei vorgegeben werden können sollen, in einer Weise erfolgen, dass das Framework eindeutig auf diese zugreifen kann. Um dies zu garantieren, werden entsprechende Daten als Schlüssel-Wert-Paare gespeichert. So kann das Framework in einem willkürlich geordneten Datensatz den geforderten Wert auswählen, ohne Kenntnis über die Anordnung der Daten zu besitzen.

4.1.4. Datenstruktur der Füllstruktur

Die Körper selber müssen in einer Datenstruktur vorliegen. Da bei der Optimierung (siehe Abschnitt 4.4) jeder Körper für sich betrachtet wird, ist für diese zunächst keine besondere Anforderung an die Datenstruktur zu nennen. Die Ausprägung der Füllstruktur kann ohne Filterung, wie von Wu (2018) implementiert, balanciert werden, indem die Teilungsebene auf die größte Ausdehnung eines Körpers begrenzt wird (siehe Abschnitt 4.4.3). Somit ist es nicht nötig, eine komplexe Datenstruktur zu verwenden, weshalb die Umsetzung mit einem eindimensionalen Array erfolgt, durch welches das Framework iteriert.

4.1.5. Nachteile gegenüber traditionellen Füllstrukturen

Der Vergleich mit einer traditionellen Füllstruktur, wie in Abschnitt 2.2.1 beschrieben, zeigt auch Nachteile der modifizierbaren Füllstruktur, welche allerdings vor allem softwareseitig sind. Die höhere Berechnungszeit aufgrund der gesteigerten Komplexität fällt bei der Erstellung der Werkzeugpfade ins Gewicht[1]. Diese können jedoch für die Herstellung mehrerer Exemplare des Modells wiederverwendet werden, sodass es sich um eine einmalige Investition handelt. Unter Umständen kann die Herstellungszeit des Werkstücks ansteigen, wenn die Füllstruktur im Zuge der Optimierung eine komplexere Ausprägung annimmt als die herkömmliche.

Wie von Podrouzek u. a. (2019) beschrieben wird, kann eine optimierte Formgebung der Füllstruktur-Elemente in Hauptbelastungsrichtung mechanisch schwächer sein als eine extrudierte, zweidimensionale Füllstruktur. Diese Feststellung bezieht sich jedoch nur auf die Gestaltung der zugrundeliegenden Geometrie ohne Optimierung auf wirkende Belastungen und trifft also nicht auf die gewählte Umsetzung zu, deren Elemente zudem näher an klassischen Füllstrukturen orientiert sind. Wird eine Optimierung auf Belastungen vorgenommen, kann eine optimierte Füllstruktur höhere Lasten aufnehmen, wie von Wu u. a. (2016), Wu u. a. (2018), Wu (2018) und Gopsill u. a. (2019) bewiesen wird.

4.2. Schnittstelle für Optimierungsvorschriften

Das Framework soll einen flexiblen Einsatz ermöglichen, um Füllstrukturen auf verschiedene vom Programmierer vorgegebene Ziele hin zu optimieren. Ihm sind daher im Auslieferungszustand weder Eigenschaften der Füllstruktur, noch Optimierungsziele oder dazu benötigte

[1] Ohne Betrachtung der Berechnungszeit für die Optimierung, die bei der herkömmlichen Füllstruktur nicht bzw. nur auf rudimentäre Weise durchgeführt wird (siehe auch Abschnitt 2.3).

Berechnungen bekannt. Es wird daher eine Schnittstelle benötigt, welche die Fähigkeit bietet, dem Framework Optimierungsvorschriften (siehe Abschnitt 3.2.3) zu übermitteln.

Die Schnittstelle muss eine beidseitige Kommunikation erlauben, da vom Programmierer implementierte Optimierungsvorschriften Daten aus der Füllstruktur wie auch aus dem Optimierungsdatensatz verarbeiten sollen. Diese werden demnach als Software-Objekte ausgelegt, welche Methoden besitzen, um Daten anzufordern, zu verarbeiten und Verarbeitungsergebnisse zurückzumelden. Ein Programmablauf für die Kommunikation zwischen Framework und Optimierungsvorschrift ist in Abb. 4.2 dargestellt.

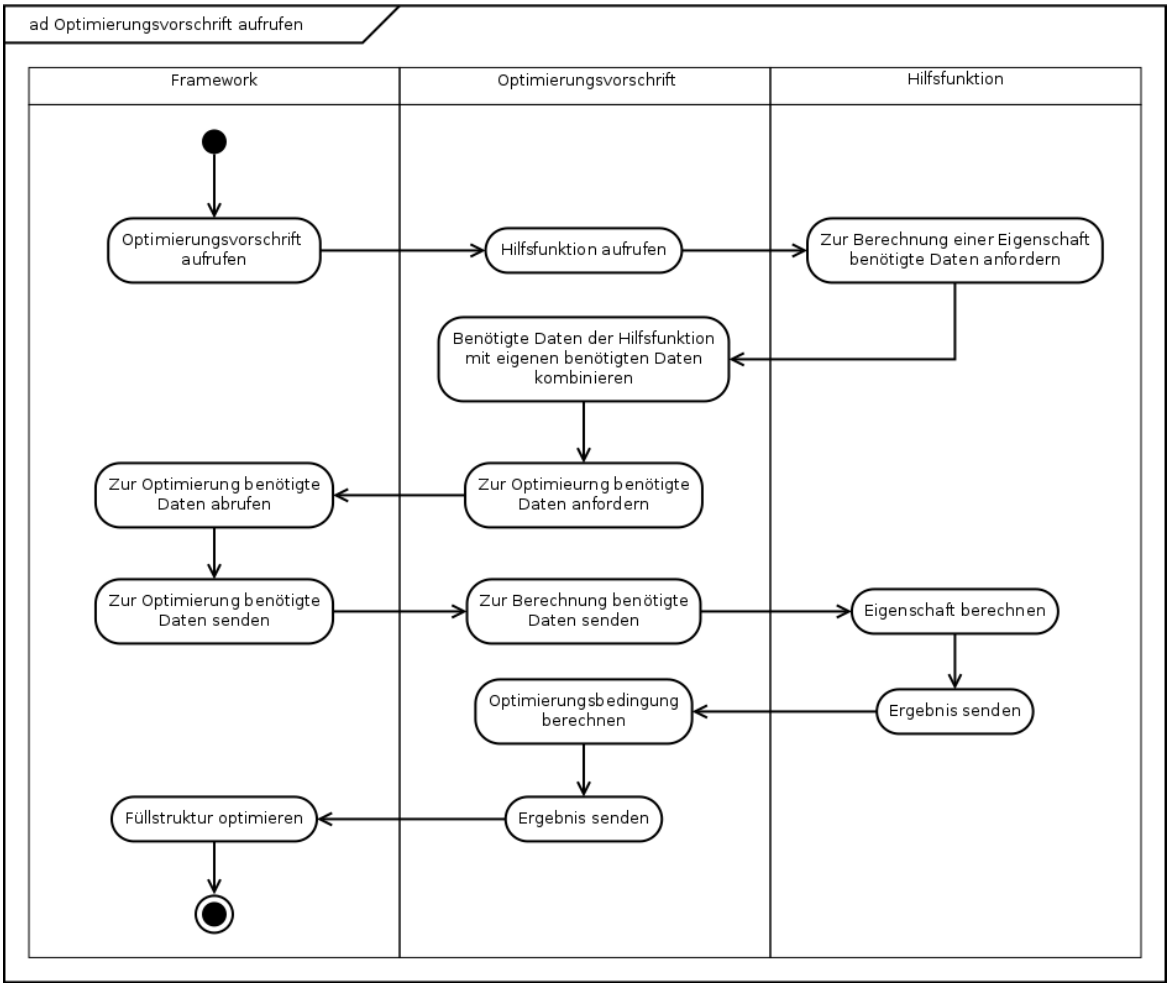


Abbildung 4.2.: Schnittstelle zwischen Framework und Optimierungsvorschrift. Der Aufruf von Hilfsfunktionen wird in Abschnitt 4.3 behandelt.

Im Optimierungsprozess wird ein Körper geladen und dessen Eigenschaften vom Framework untersucht. Wie in Abb. 4.2 illustriert, ruft das Framework die entsprechende Vorschrift

zu einer Eigenschaft auf. Das Framework hat an dieser Stelle noch keine Kenntnis über weitere benötigte Daten, daher muss die Berechnungsvorschrift über die in Abschnitt 3.2.3 erwähnten Instruktionen Daten vom Framework anfragen.

4.2.1. Instruktionen

Die Instruktionen müssen so gestaltet sein, dass das Framework beliebigen Optimierungsvorschriften die von ihnen benötigten Daten übergeben kann, solange diese vorhanden sind. Die Eigenschaften von Körpern der Füllstruktur wie auch des Optimierungsdatensatzes werden vom Framework als Schlüssel-Wert-Paare gespeichert. Daher ist es sinnvoll, die Instruktionen so zu gestalten, dass sie dem Framework die entsprechenden Schlüssel übergeben, sodass es die dazugehörigen Werte abrufen kann.

Dieser Ansatz ist in der Hinsicht flexibel, dass beliebige Daten, dessen Schlüssel bekannt sind, abgerufen werden können. Der implementierende Programmierer muss an dieser Stelle demnach neben von ihm definierten Eigenschaften im Optimierungsdatensatz nur Kenntnis über vom Framework eingesetze Schlüssel haben. Diese müssen daher in der Dokumentation des Frameworks definiert sein. Außerdem müssen die Instruktionen die für die Optimierung benötigte Teilungsebene (siehe Abschnitt 4.4.3) beinhalten.

4.2.2. Übergabe der benötigten Daten

Die abgerufenen Daten werden vom Framework der Optimierungsvorschrift als Schlüssel-Wert-Paare bereitgestellt, sodass ein durchgängiges Identifizierungsschema der Daten gewährleistet wird. Bei der Implementierung der Berechnung (siehe 4.2.3) hat dies den weiteren Vorteil, dass verwendete Daten auch für den Programmierer eindeutig und aussagekräftig referenzierbar sind. Ein solcher Ansatz reduziert Programmierfehler, da Daten mit der Eigenschaft, die sie repräsentieren sollen (siehe auch Abschnitt 3.2.2), assoziiert werden.

4.2.3. Berechnung

Um Optimierungsziele mit dem aktuellen Zustand der Füllstruktur zu vergleichen, müssen Berechnungen getätigt werden, die nicht zwangsläufig einfache Vergleiche von Ist- und Sollzustand sind. Einige Eigenschaften, wie beispielsweise die durchschnittliche Dichte eines Körpers, müssen aus anderen abgeleitet werden, wozu weitere Schritte (siehe auch 4.3) nötig sind.

Da Inhalte des Optimierungsdatensatzes frei vorgegeben werden können, ist es daher ebenfalls denkbar, dass diese keine Eigenschaften, sondern auf die Füllstruktur wirkende externe Faktoren darstellen. Die Optimierungsvorschrift muss in solchen Fällen die Reaktion der Füllstruktur auf diese Faktoren berechnen, damit das Framework eine Optimierung der betroffenen Eigenschaften vornehmen kann.

Ein Vorteil davon, diese Berechnungen in der Optimierungsvorschrift durchzuführen, ist, dass der Programmierer der Vorschrift auch den Aufruf von externen Programmen einsetzen kann, um deren Funktionalität zu nutzen. Beispielsweise könnte eine FEM-Analyse für bestimmte Optimierungen nötig sein, zu welcher in der Optimierungsvorschrift die nötige Schnittstelle implementiert wird.

Optimierungsvorschriften müssen eine Methode besitzen, mit der die Berechnung durchgeführt wird. Diese erhält die vom Framework bereitgestellten Daten und gibt den unten beschriebenen Rückgabewert zurück.

4.2.4. Rückgabewert

Über die erfolgte Berechnung wird die Notwendigkeit der Modifikation eines Körpers zur Optimierung der Füllstruktur ermittelt. Da die Entscheidung für eine Optimierung binär ist, muss der Rückgabewert der Berechnung vom Typ `boolean` sein. Durch die Umsetzung der Schnittstelle mithilfe eines einzelnen Datentyps kann das Framework zudem beliebige Optimierungsvorschriften ohne weiteren Programmieraufwand einbinden und verwenden.

4.3. Hilfsfunktionen

Wie in Abschnitt 3.2.4 beschrieben, sind einige Eigenschaften nicht unmittelbar abzurufen. Die Berechnung innerhalb einer Optimierungsvorschrift (Abschnitt 4.2.3) kann auf ein solches Problem stoßen. Prinzipiell sind hier zwei Lösungswege möglich: Das Konstruieren der abgeleiteten Eigenschaft vor der Übergabe an die Optimierungsvorschrift oder die Übergabe der konstituierenden Eigenschaften an die Optimierungsvorschrift und das Bilden der abgeleiteten Eigenschaft dort.

Der erste Lösungsweg ermöglicht eine saubere Trennung der Funktionen. Allerdings ist der Programmablauf komplex und kann somit zu einer höheren Fehleranfälligkeit beim Einsatz des Frameworks führen, da der Programmierer der Optimierungsvorschrift zusätzlich eine Hilfsfunktion an der entsprechenden Stelle ins Framework einbinden muss.

Wird die abgeleitete Eigenschaft innerhalb der Optimierungsvorschrift berechnet, muss diese die konstituierenden Eigenschaften vom Framework anfordern. Der Programmablauf hierfür

wird in Abb. 4.2 dargestellt. Eine Hilfsfunktion ist zwar dafür nicht notwendig, jedoch sinnvoll, um eine hohe Wiederverwendbarkeit des Codes zu erreichen[2]. Der Datenfluss ist mit diesem Ansatz weniger komplex und besser überschaubar. Daher werden in der Umsetzung des Frameworks Hilfsfunktionen von den Optimierungsvorschriften aufgerufen.

4.4. Algorithmus

Der Optimierungsalgorithmus muss Daten von Füllstruktur, Optimierungsdatensatz und Optimierungsvorschriften zusammenbringen, um die Optimierung der Füllstruktur zu steuern. Wie Abb. 4.3 zeigt, fragt das Framework dabei sequentiell von jeweiligen Stellen Daten ab. In der gewählten Implementierung reagiert die Füllstruktur ausschließlich auf lokale Vorgaben aus dem Optimierungsdatensatz. Wie Abschnitt 4.1.4 angedeutet, iteriert der Algorithmus dafür durch die Füllstruktur, sodass immer nur ein Körper zur Zeit optimiert wird.

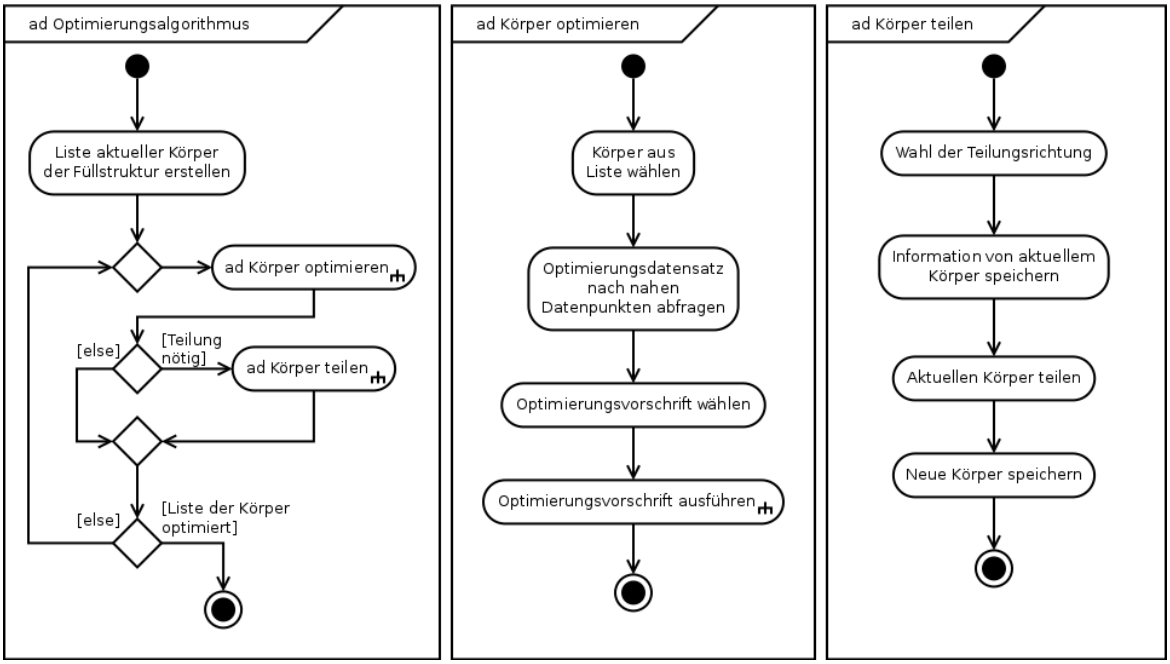


Abbildung 4.3.: ADs des Optimierungsalgorithmus'. Das AD "Optimierungsvorschrift ausführen" wird in Abb. 4.2 beschrieben.

[2] Wenn der Aufbau von Hilfsfunktionen zudem standardisiert ist, kann eine Schnittstelle implementiert werden, die es erlaubt, Hilfsfunktionen in das Framework einzubinden. Dies wird an dieser Stelle jedoch nicht weiter betrachtet.

Die Betrachtung der Nachbarn oder der Hierarchie ist daher nicht nötig. Sie kann jedoch in weiteren Ausarbeitungsstufen dann sinnvoll sein, wenn beispielsweise Strukturen über mehrere Körper hinweg erschaffen werden sollen (siehe auch Abschnitt 3.1.6).

4.4.1. Auswählen von Datenpunkten

Mit den Positions- und Dimensionsdaten des geladenen Körpers können Datenpunkte aus dem Optimierungsdatensatz gewählt werden, die für diesen relevant sind. Datenpunkte können innerhalb (inklusive Grenzfälle) und außerhalb eines Körpers situiert sein. Siehe dazu Abb. 4.4. Bei der Wahl der Datenpunkte sind jene innerhalb eines Körpers zu bevorzugen, schließlich repräsentieren sie die (gewünschten) Eigenschaften des vom Körper eingenommenen Raumes.

Es kann beispielsweise durch eine geringe Auflösung des Optimierungsdatensatzes der Fall eintreten, dass kein Datenpunkt innerhalb eines Körpers liegt. Hier ist es sinnvoll, denjenigen zu wählen, der am nächsten zum Mittelpunkt des Körpers situiert ist. Werden mehrere Datenpunkte mit der gleichen Distanz gefunden, wird der erste gewählt. Ein solcher Fall sollte jedoch durch eine genügend hohe Auflösung des Optimierungsdatensatzes vermieden werden. Eine Erweiterung kann sein, eine Lösung durch Interpolation mehrerer Datenpunkte zu erzeugen.

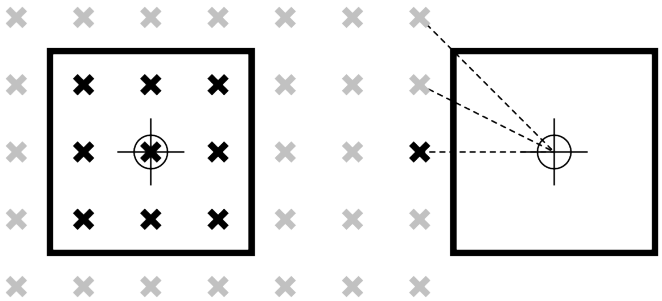


Abbildung 4.4.: Wahl von Datenpunkten innerhalb (links) bzw. am nächsten zum Mittelpunkt (rechts) eines Körpers

Einige Optimierungen maximieren Eigenschaften, während andere diese zu minimieren versuchen. Außerdem können Eigenschaften über das gesamte Volumen des Körpers betrachtet werden, sodass eine Mittelwert- oder Medianbildung ebenfalls in Frage kommen. Daher muss bei der Optimierung einer Eigenschaft die Option gegeben werden, die entsprechenden Daten aus der Auswahl der Datenpunkte auszuwählen.

4.4.2. Optimierungsvorschrift

Der Optimierungsdatensatz kann mehrere Optimierungsziele enthalten, deren Umsetzung miteinander konkurrieren. Die umzusetzende Optimierungsvorschrift muss daher aus der dem Framework übergebenen Liste der Vorschriften ausgewählt werden, sodass regional verschiedene Optimierungen vorgenommen werden können. Wie in Abschnitt 4.2 beschrieben, übergibt eine Optimierungsvorschrift nach Aufruf dem Framework Instruktionen. Dieses ruft daraufhin die zur Optimierung benötigte Methode der Optimierungsvorschrift mit den entsprechenden Daten auf und entscheidet anhand des Rückgabewertes, ob eine Modifikation des aktuellen Körpers nötig ist.

4.4.3. Bestimmen der Teilungsebene

Bei der Modifikation der Körper durch Teilung muss eine Teilungsebene bestimmt werden. Diese kann, wie Abb. 4.5 zeigt, drei Ausrichtungen annehmen, wenn die Ebene parallel zu den Körperwänden angeordnet wird.

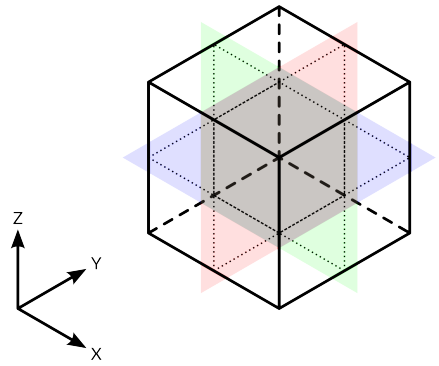


Abbildung 4.5.: Teilungsebenen parallel zur XY-Ebene (blau), XZ-Ebene (grün) und YZ-Ebene (rot)

Um die Teilungsebene zu ermitteln, wird der Gradient der zu optimierenden Eigenschaft aus den Datenpunkten gebildet. Die Teilungsebene kann dabei parallel oder orthogonal zum Gradienten der Eigenschaft, die optimiert werden soll, verlaufen (Abb. 4.6).

Ausrichtung der Teilungsebene

In der Regel ist es sinnvoll, Körper orthogonal zu Eigenschaftsgradienten zu teilen. Soll beispielsweise die Füllstruktur an Körpergröße abnehmen, je näher sie der Modellaußenwand

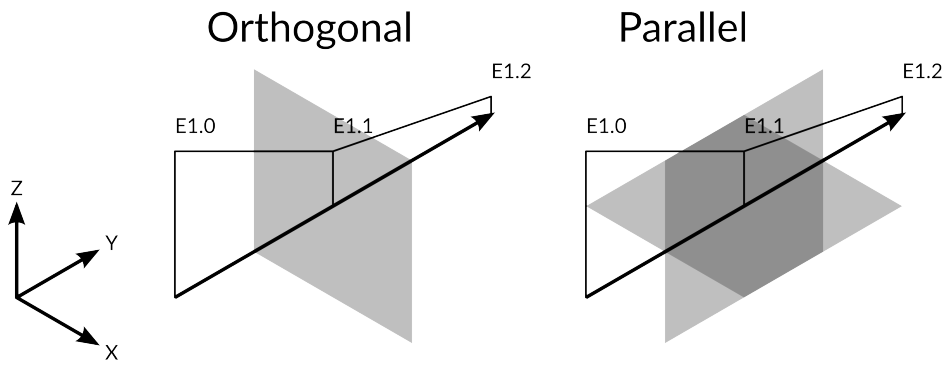


Abbildung 4.6.: Teilungsebenen parallel bzw. orthogonal zum Eigenschaftsgradienten (E1.0, E1.1, E1.2)

ist, ist eine orthogonale Teilung gegeben, da dadurch die Größe der Unterkörper in Richtung des Distanz-Gradienten abnehmen kann. Die Optimierung auf Druckfestigkeit ist ebenfalls in der Lage, die Füllstruktur durch die zusätzliche Wand parallel zu wirkenden Kräften (und somit zum Gradienten der Kräfte) in deren Richtung widerstandsfähiger zu gestalten. Siehe Abb. 4.7 für eine Illustration beider Beispiele. Zum Zeitpunkt der Ausarbeitung liegen keine Beispiele von Eigenschaften vor, die von einer parallelen Teilungsebene profitieren. Allerdings soll dieser Fall nicht ausgeschlossen werden und wird in der Umsetzung also als Option mit berücksichtigt.

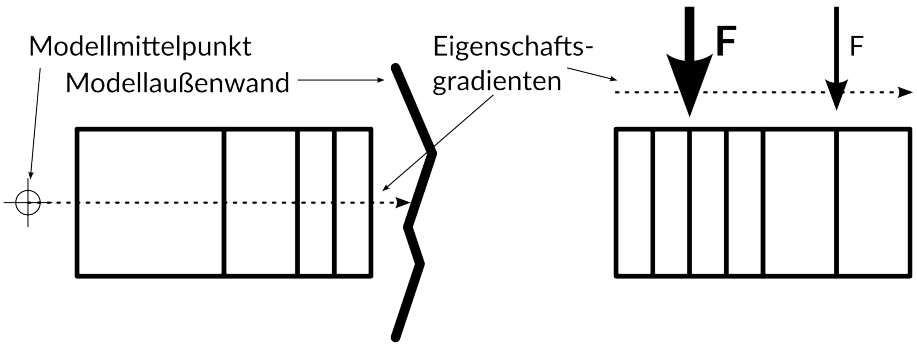


Abbildung 4.7.: Teilung orthogonal zur Distanz zur Außenhülle (links), parallel zu wirkenden Kräften (rechts)

Optionale Reduzierung der Isotropie

Es werden teilweise stark isotrope Strukturen (Abb. 4.8) erzeugt, wenn eine Region von einem Optimierungsziel und einem Gradienten dominiert wird. Wie Wu (2018) und Wu u. a. (2018) beschreiben, führt eine hohe Isotropie zwar zu einer guten Anpassung an ein Optimierungsziel, aber auch zu einer hohen Anfälligkeit für unvorhergesehene Einwirkungen. Es wird daher eine Option implementiert, mit der der Schlankheitsgrad der Körper beschränkt und so mit ausgeglichenen Körpern eine gewisse Orthotropie hergestellt werden kann.

Zusätzlich erstrecken sich schlanke Körper über eine große Region und können somit Datenpunkte aus verschiedenen Bereichen des Optimierungsdatensatzes einschließen. Dies kann sich in der Anwendung negativ auswirken, da eine lokale Optimierung dadurch immer schwieriger wird. Wenn aus den Datenpunkten ein Eigenschaftswert ausgewählt werden soll, könnten zudem Daten, die viel weiter vom Körpermittelpunkt entfernt sind, relevantere, nähere Daten in der Auswahl verdrängen. Abb. 4.8 zeigt ein solches Problem, wo ein betracheter Datenpunkt (rotes X) nur noch eine kleine Region in einem weit ausgebreiteten Körper repräsentieren kann. Dagegen kann eine Füllstruktur mit ausgeglichenen Körpern lokal wesentlich eingeschränktere Optimierungen erzeugen. Man beachte, dass die Fläche der kleinsten Körper in beiden Füllstrukturen gleich ist.

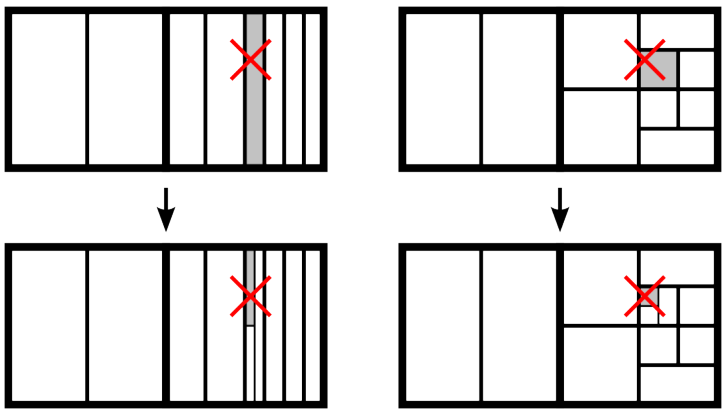


Abbildung 4.8.: Vergleich zwischen Füllstruktur aus schlanken (links) und ausgeglichenen (rechts) Körpern

4.5. Eingabedatensatz

Grundeinstellungen und Optimierungsziele sind strukturell ähnlich aufgebaut (siehe Abschnitte 3.2.1 und 3.2.2). Daher besteht die Möglichkeit, diese zu einem Eingabedatensatz zusammenzufassen. Weiterhin definieren die Grundeinstellungen beispielsweise den Optimierungsraum und Werkstoffdaten, die mit einem Optimierungsdatensatz verknüpft sind. Aufgrund dessen wird im Rahmen dieser Arbeit zunächst ein Eingabedatensatz mit Grundeinstellungen und Optimierungsdatensatz implementiert. Eine spätere Revision sollte jedoch nicht ausgeschlossen sein, wenn bei der weiteren Entwicklung des Frameworks und der weiteren Software die Nötigkeit erkannt wird, beide Datensätze voneinander zu trennen.

4.5.1. Wahl des Datenformates

Mit den gegebenen Anforderungen aus Abschnitten 3.2.1 und 3.2.2 wird JSON als das Datenformat für den Eingabedatensatz gewählt. Wenn das Format von Menschen lesbar sein soll, fällt MessagePack weg, obwohl es vielversprechende Features aufweist. Gerade die kompakte Formatierung der Daten wie auch die Möglichkeit des Speicherns von binären Daten heben dieses Format von anderen ab. XML ist bezüglich der Lesbarkeit und kompakten Darstellung der Daten gegenüber JSON und YAML im Nachteil. Ein Vorteil von YAML gegenüber JSON ist die Möglichkeit, selbst definierte Datentypen zu speichern. Eine bessere Lesbarkeit ist ebenfalls zu nennen. Dagegen ist die Implementierung von JSON schneller in der Datenverarbeitung, was gerade bei großen Datensätzen vorteilhaft ist [3] [4]. Die ambiguose und komplexe Spezifikation von YAML kann zudem zu Fehlern und unerwarteten Verhalten bei der Interpretation von Daten führen[5][6]. Um eine robuste Umsetzung mit geringer Wahrscheinlichkeit von durch das Datenformat erzeugten Bugs zu erreichen, ist JSON daher zu bevorzugen.

4.5.2. Grundeinstellungen

Die Grundeinstellungen beinhalten wie in Abschnitt 3.2.1 dargelegt den Optimierungsraum und die Werkstoffdaten des eingesetzten Rohmaterials. Der Optimierungsdatensatz und das 3D-Modell können an verschiedenen Koordinatensystem ausgerichtet sein[7], weshalb der

[3] https://codersbuffet.blogspot.com/2010/03/json-vs-xml-and-python-parsing.html
[4] https://gist.github.com/havenwood/4513627
[5] https://arp242.net/yaml-config.html
[6] https://github.com/cblp/yaml-sucks/blob/master/README.md, abgerufen: Commit 99b61b5
[7] Im Rahmen dieser Arbeit sind beide Koordinatensysteme identisch.

Koordinatenursprung des Optimierungsraumes ebenfalls in den Grundeinstellungen gespeichert.

Außerdem wird das Gitter des Optimierungsdatensatzes mit dessen Auflösung und maximalen Index beschrieben. Dies erlaubt es dem Framework, die Position eines Datenpunkts innerhalb des Datensatzes zu bestimmen (siehe nächsten Abschnitt).

Damit das Framework die Füllstruktur initialisieren kann, wird zuletzt die anfängliche Ausprägung der Füllstruktur in den Grundeinstellungen übermittelt. Da weitere Parameter wie der Optimierungsraum bereits Aspekte der Füllstruktur beschreiben, wird im Prototyp nur die anfängliche Körpergröße benötigt, um die Füllstruktur vollständig zu definieren.

4.5.3. Optimierungsdatensatz

Die Datenpunkte liegen als dreidimensionales, gleichmäßiges Gitter vor und müssen eindeutig identifizierbar sein (siehe auch Abschnitt 3.2.2). Das gleichmäßige Gitter, das sich aufgrund der übersichtlicheren Struktur für den Prototyp anbietet, kann zu einem späteren Zeitpunkt durch eines variabler Dichte (siehe Abschnitt 3.2.2) ersetzt werden. Damit der Schlüssel eines Datenpunktes gleichzeitig als Positionsmarke im Gitter dienen kann, wird dieser aus den Indizes in der jeweiligen Gitterdimension zusammengesetzt. Ein Datenpunkt `"01.02.03":{[...]}` belegt somit entsprechende Indizes 1, 2 bzw. 3. Somit kann das Framework unter Zuhilfenahme der in den Grundeinstellungen übermittelten Informationen zum Optimierungsdatensatz auch ohne Suchen gezielt auf den lokalen Kontext eines Datenpunkts zugreifen.

Die alternativ mögliche Identifizierung über die räumliche Position wird nicht umgesetzt, da die Raumkoordinaten nicht zwangsläufig reelle Zahlen sind. Eine eindeutige In- bzw. Dekrementierung wie bei einem `int` wäre somit nicht möglich. Diese werden stattdessen zusätzlich im Datenpunkt gespeichert.

Neben den Metadaten werden die lokalen Optimierungsziele im Datenpunkt gespeichert. Die Benennung ihrer Schlüssel muss eindeutig und möglichst deskriptiv, wie auch mit deren Benennung in den Instruktionen der Optimierungsvorschriften identisch sein.

4.6. Ausgabe

Die Ausgabe des Frameworks soll von weiteren Programmteilen verarbeitet werden, weshalb sie nicht für eine visuelle Darstellung ausgelegt ist. Im Rahmen dieser Arbeit wird nur das Framework umgesetzt, sodass für das Debugging und das Präsentieren von Ergebnissen zusätzlich eine Visualisierung von Schichtinformationen implementiert wird.

4.6.1. Wahl des Formates für die Ausgabe der Füllstruktur

Die Überlegungen zum Optimierungsdatensatz (siehe Abschnitt 4.5.3 können mit wenigen Veränderungen für die Ausgabe der Füllstruktur übernommen werden, da die Datensätze strukturell ähnlich sind. Daher kann für die Füllstruktur das gleiche Datenformat verwendet werden (siehe Abschnitt 4.5.1).

Eine Umsetzung des Frameworks unter Verwendung möglichst weniger verschiedener Datenformate hat Vorteile wie die Einbindung weniger Bibliotheken und damit weniger Abhängigkeiten. Programmierer, die das Framework einsetzen möchten, werden zudem vor einen geringeren Aufwand gestellt, eventuell neue Konzepte zu erlernen.

4.6.2. Visuelle Ausgabe

Für die Umsetzung der visuellen Ausgabe wird das Scalable Vector Graphics (SVG) Format gewählt, um Schichtinformationen zu exportieren. Da dieses auf dem XML-Standard basiert, können Graphiken über Standardbibliotheken erstellt werden. Die Schichtinformationen liegen als Pfade vor, eine verlustfreie beliebige Vergrößerung erlaubt also das Betrachten von kleinsten Details.

Export-Tool

Ein Export-Tool wird entwickelt, um Schichtinformationen im SVG-Format zu speichern. Dieses iteriert durch die Füllstruktur und leitet aus der Geometrie der einzelnen Körper Vektorpfade ab, die ebenenweise nach Z-Koordinaten sortiert werden. Wenn eine Wandfläche parallel zur XY-Ebene ausgerichtet ist, wird vom Export-Tool eine Fläche in der SVG-Datei eingesetzt[8]. Dabei wird die Wandfläche, wie in Abb. 4.9 dargestellt, der nächst kleineren Z-Koordinate zugeordnet[9].

Zuschnitt

Die Füllstruktur ist bei der Ausgabe noch nicht auf das 3D-Modell zugeschnitten. Es kann sich des Masken-Features des SVG-Formates bedient werden, um einen visuellen Beschnitt

[8] Eine einfache Erweiterung für Harz- und Pulververfahren wäre das Einsetzen einer Aussparung Lochs in der Fläche, sodass das Abfließen des Rohmaterials gewährleistet ist. Da sowohl die obere wie auch die untere Wandfläche damit durchlässig ist, entsteht zudem kein das Rohmaterial zurückhaltender Unterdruck.

[9] Bei herkömmlichen Slicern wird ein ähnlicher Ansatz realisiert. Da die Dicke der Extrusionsspur der Schichtdicke entspricht, ist eine Toleranz von maximal einer Schichtdicke gegeben.

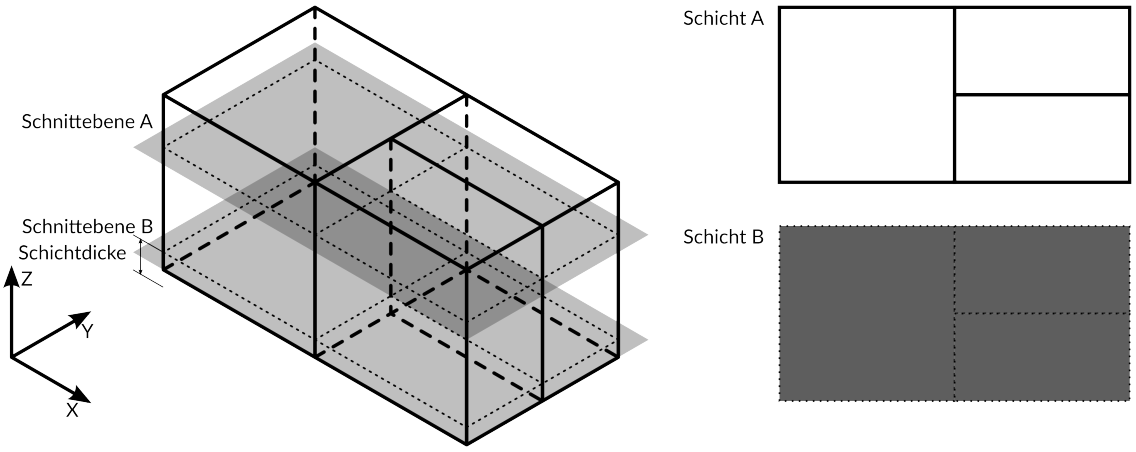


Abbildung 4.9.: Export und Zuordnung von Füllstrukturgeometrie zu Schichten

für das Debugging umzusetzen, indem die Sichtbarkeit von Graphiken durch maskierende Graphiken gesteuert wird[10], wie Abb. 4.10 zeigt. Die Transparenz der maskierten Graphik wird dabei vom Helligkeitswert der Maske von Schwarz für vollständig transparent bis Weiß für vollständig opak eingestellt.

Abbildung 4.10.: Maskieren von Graphiken: Zu maskierendes rotes Rechteck (links) wird mit einer Graustufen-Maske versehen (mitte), sodass seine Sichtbarkeit modifiziert wird (rechts).

Slicing des 3D-Modells

Wie oben beschrieben, werden die Schichtinformationen der zugehörigen 3D-Modells benötigt, um die Darstellung der Füllstruktur zu beschneiden. Für eine einfache Umsetzung im Rahmen dieser Arbeit kann der SVG-Export von Slic3r verwendet werden, um Schichtinformationen zu extrahieren[11], welche vom Export-Tool interpretiert werden. Schichten werden

[10] https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Clipping_and_masking
[11] https://manual.slic3r.org/advanced/svg-output

jeweils als Maske für die entsprechende Schicht in der Füllstruktur eingestellt. Dabei wird sich der Belichtungseigenschaft des SLA-Prozesses auf DLP-Basis (siehe Abschnitt 2.1.2) bedient: Zu belichtende Sektionen (Modell) sind weiß, während andere Bereiche entweder keine Graphik enthalten oder schwarz sind (siehe Abb. 4.11). Dies korrespondiert mit den Einstellungen für Masken im SVG-Format, sodass keine weitere Änderung vorgenommen werden muss.

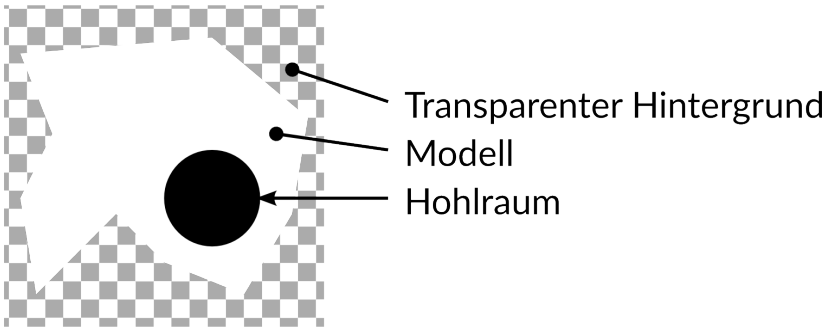


Abbildung 4.11.: Darstellung einer SVG-Schichtausgabe von Slic3r

Dateiinhalt und Darstellung

Das SVG-Format unterstützt das Speichern mehrerer Graphiken in einer Datei, doch die meisten Programme zur Anzeige von SVG-Daten sind nicht darauf ausgelegt, diese einzeln darzustellen. Daher müssen Schichten entweder als separate Dateien gespeichert werden. Alternativ kann ein spezielles Programm[12] verwendet werden, welches dafür konzipiert ist, die Slicer-Ausgabe für SLA-Drucker darzustellen, welche alle Schichten als Gruppen[13] in einer einzelnen Datei beinhaltet.

4.7. Testdateien

Durch das Erzeugen eines Optimierungsdatensatzes, der mindestens zwei Regionen besitzt (siehe Abb. 4.12), in denen sich die Optimierungsvorgaben einer Eigenschaft stark unterscheiden, können mehrere Kriterien überprüft werden. Mit den distinkten Regionen kann verifiziert werden, dass mehrere Ausprägungen einer Optimierungsvorgabe möglich sind.

[12] http://garyhodgson.github.io/slic3rsvgviewer/
[13] https://developer.mozilla.org/en-US/docs/Web/SVG/Element/g

Da die Füllstruktur, wie in Abschnitt 3.1.6 beschrieben, Eigenschaftsgradienten näherungsweise wiedergeben soll, muss zusätzlich mindestens ein Optimierungsdatensatz zur Bildung eines solchen Gradienten führen.

Wird der Optimierungsdatensatz transformiert, muss die Füllstruktur dieser Transformation folgen. Wie Abb. 4.12 zeigt, folgt auf einer Rotation des Optimierungsdatensatzes ebenso eine Rotation der Füllstruktur. Hiermit kann die korrekte Ausrichtung wie auch die Generalisierbarkeit der Füllstruktur auf verschiedene Optimierungsvorgaben (zunächst einer Eigenschaft) verifiziert werden.

Ein Überschreiten der Optimierung (fehlendes Beenden der Optimierung) kann festgestellt werden, indem die Optimierungsvorgabe einen bestimmten Wert vorgibt, der zu Ausprägungen zwischen den maximalen Ausprägungen der Körper führt. Die erreichte Ausprägung wird dann mit der geplanten verglichen. Das Einhalten von Grenzbedingungen ist daran zu erkennen, dass Körper die minimale bzw. maximale Größe nicht unter- bzw. überschreiten.

Eine Anwendbarkeit von mehreren Optimierungszielen kann auf gleiche Weise überprüft werden. Diese müssen dafür eindeutig voneinandner unterscheidbare Ausprägungen der Füllstruktur bewirken und verschiedene Regionen im Optimierungsdatensatz einnehmen.

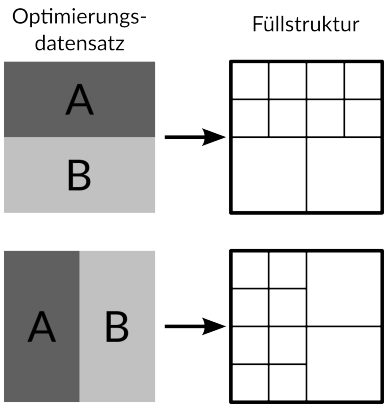


Abbildung 4.12.: Testdateien mit zwei unterschiedlichen Regionen (A/B) für die Optimierung muss verschiedene Füllstruktur-Ausprägungen erzeugen.

5. Realisierung

Für die Umsetzung des Framework-Prototyps wird die Programmiersprache Python gewählt. Python ist für diese Aufgabe besonders gut geeignet, da es die Scriptsprache durch ein extensives Angebot an Softwarebibliotheken erlaubt, schnell einen funktionsfähigen Softwareprototypen zu erstellen[1]. Zudem ist die Sprache etabliert und beliebt[2][3], was eine hohe Zahl an potentiellen Entwicklern mit sich führt. Im Vergleich zu einer kompilierten Sprache ist Python wesentlich langsamer[4], was im Prototypenstadium jedoch einer geringeren Relevanz beigemessen werden sollte, zumal mit Optimierungen eine höhere Verarbeitungsgeschwindigkeit möglich ist[5].

Es wird ein Software-Prototyp erstellt, der alle nötigen Elemente des Frameworks enthält. Diese werden vom beigefügten Testprogramm aufgerufen, um Ausgaben zur Validierung der Software zu erstellen. Da sich das Framework noch in einem sehr frühen Prototypenstadium befindet, wird dieses noch nicht in einer als Paket gebündelten Form publiziert.

5.1. Engine

Die Klasse `Engine` stellt alle Funktionalität bereit, mit der die Füllstruktur erzeugt und verändert werden kann. Sie bildet die zentrale Komponente des Frameworks, welche den Datenfluss steuert.

[1] https://stxnext.com/python-vs-other-programming-languages/#why-python
[2] https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/
[3] https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted
[4] https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/python3-gcc.html
[5] https://www.ibm.com/developerworks/community/blogs/jfp/entry/A_Comparison_Of_C_Julia_Python_Numba_Cython_Scipy_and_BLAS_on_LU_Factorization?lang=en

5.1.1. Initialisierung

Bei der Initialisierung des Frameworks wird der Instanz von `Engine` das Container-Objekt des Eingabedatensatzes übergeben. Aus diesem werden alle nötigen Informationen aus den Grundeinstellungen extrahiert, um die anfängliche Ausprägung der Füllstruktur zu erzeugen. Die Methode `create_cell_structure()` baut diese dann nach den Vorgaben auf.

5.1.2. Algorithmus

Der Optimierungsalgorithmus (Abschnitt 4.4) wird durch Methoden der Klasse `Engine` implementiert. Durch das Umstellen des Status-Flags bei Erreichen des gewünschten Zustandes können Körper nach einer Iteration der Optimierung von dieser ausgeschlossen werden (siehe Abschnitt 5.2.2). Dazu werden sie in ein weiteres Array, welches ausschließlich Körper im finalen Zustand enthält, übertragen. Außerdem wird eine Option implementiert, mit der die Anzahl der Optimierungs-Iterationen beschränkt werden kann. Die Füllstruktur lässt sich so nach jedem Optimierungsschritt betrachten.

5.1.3. Modifikation eines Körpers

Bei der Modifikation (Teilung) eines Körpers werden zunächst seine Attribute ausgelesen und dieser dann gelöscht. Lage und Dimensionen der neuen Unterkörper werden berechnet und die Attribute des ursprünglichen Körpers übertragen. Der erste Unterkörper erhält die ID des ursprünglichen Körpers, während der zweite eine neue ID zugewiesen bekommt, die durch die Inkrementierung der höchsten Körper-ID berechnet wird.

5.2. Füllstruktur

Die Füllstruktur wird als Array von Software-Objekten, welche die einzelnen Körper darstellen, umgesetzt. Somit ist lediglich die Umsetzung eines einzelnen Körpers zu spezifizieren.

5.2.1. Eigenschaften

Jedes Objekt besitzt Attribute, korrespondierend mit Metadaten und Eigenschaften des Körpers. Diese werden bei der Initialisierung zugewiesen bzw. berechnet und werden folgendermaßen unterteilt:

- ID
- Ort und Geometrie
- Zugewiesene Eigenschaften (Werkstoff, Wandstärke etc.)
- Status-Flag (Optimierung abgeschlossen / nicht abgeschlossen)

Der Zugriff auf einzelne Kategorien kann somit unabhängig von den anderen Attributen geschehen.

5.2.2. Interaktion mit Eigenschaften

Jedes Objekt besitzt weiterhin Methoden, mithilfe derer auf Attribute zugegriffen werden können. Dabei können alle Attribute gelesen, aber nur weitere Eigenschaften zugewiesen werden. Ein Verändern von Attributen ist bis auf der Status-Flag verboten. Wie oben beschrieben, werden diese bei der Initialisierung festgesetzt, sodass sie nur dann verändert werden können, wenn ein Körper modifiziert wird (siehe Abschnitt 5.1.3).

Die Status-Flag muss veränderbar sein, damit bei der Optimierung der Füllstruktur die weitere Modifikation des Körpers gestoppt werden kann, wenn eine gewünschte Ausprägung erreicht ist. Dazu wird dieses Attribut vom Typ `boolean` umgestellt, sodass der Körper in den nächsten Optimierungsschritten ignoriert wird.

5.2.3. Modifikation des Körpers

Die Modifikation eines Körpers wird vom Framework vollzogen. Hierfür besitzt ein Körper keine besonderen Methoden.

5.2.4. Implementierung der Geometrie

Geometrieelemente werden als Klassen umgesetzt. Jedes Geometrieelement besitzt Methoden zur Abfrage von Daten wie Ort und Ausbreitung des Elementes. Mit steigender Elementdimension werden abrufbare Daten entsprechend komplexer. Besitzt ein Eckpunkt nur Informationen zur Lage im Koordinatensystem, kann eine Kante bereits Länge, eigene Lage, sowie Lage der Eckpunkte beinhalten, während eine Fläche zusätzlich zu den genannten Informationen über einen Flächeninhalt verfügt.

Da die Körper aus Quadern bestehen, müssen die beinhalteten Geometrieelemente entsprechend angesprochen werden, um für Berechnungen benötigte Daten zu erhalten. Zu

diesem Zweck wird eine normierte Indizierung vorgenommen, die immer mit dem Element beginnt, deren erster Punkt am nächsten zum Koordinatenursprung liegt. Alle weiteren Elemente werden mathematisch positiv und mit steigender Z-Koordinate numerisch aufsteigend indiziert (siehe Abb. 5.1).

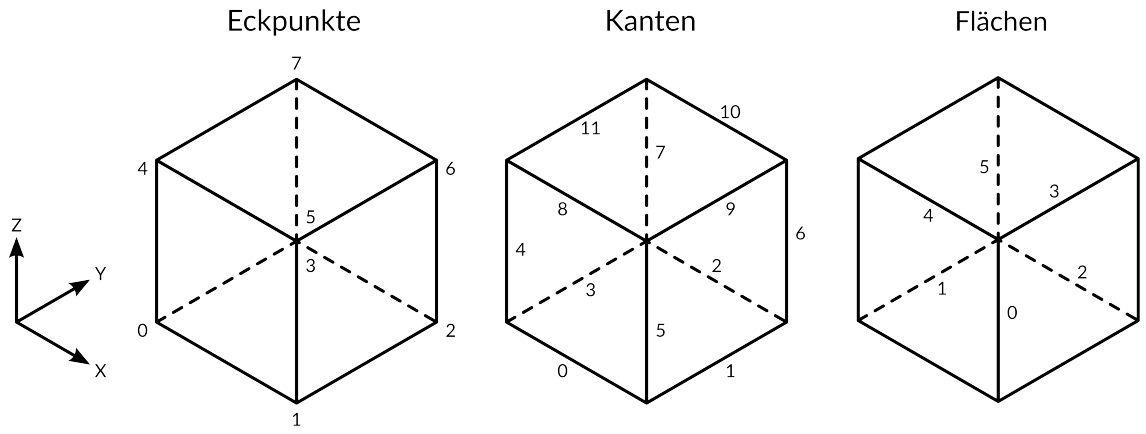


Abbildung 5.1.: Nummererierung von Geometrieelementen

Die einfache Implementierung von Geometrieelementen kann in weiteren Ausarbeitungsstufen durch vollständige Geometriebibliotheken ersetzt werden, wenn dies sich als nötig erweisen sollte. Für die Verwendung im Prototyp sind jedoch eine geringe Komplexität und fehlende Abhängigkeit von externen Bibliotheken Kriterien, die zusammen mit der einfachen Körpergeometrie die gewählte Lösung legitimieren.

5.3. Eingabedatensatz

Um dem Framework den Eingabedatensatz bereitzustellen, wird dieser in einen Container[6] geladen, welcher alle nötigen Methoden bereitstellt, um gezielt auf die in Grundeinstellungen und Optimierungsdatensatz beinhalteten Daten zuzugreifen. Durch die dadurch erreichte Kapselung kann ein eindeutiger Zugriffsweg geschaffen werden, was die unbedachte Manipulation von Daten verhintert.

[6] http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/container.html

5.3.1. Auswahl von Datenpunkten des Optimierungsdatensatzes

Wie in Abschnitt 4.4.1 beschrieben, werden Datenpunkte des Optimierungsdatensatzes den Körpern der Füllstruktur örtlich zugewiesen. Hierfür werden Suchfunktionen erstellt, die die geforderten Datenpunkte auswählen. Für die Erstellung eines Eigenschaftsgradienten der lokale Kontext eines Datenpunktes nötig, weshalb ebenso eine Auswahl nach ID erfolgt, indem der jeweile räumliche Index in einer bestimmten Schrittweite in- bzw. dekrementiert wird (siehe Abschnitt 4.5.3).

5.3.2. Erstellen eines Eingabedatensatzes

Da die Software, in die das Framework eingebettet werden soll, noch nicht existiert, müssen Eingabedatensätze durch ein eigens dafür geschriebenes Python-Script, im Folgenden `GridMaker` genannt, erzeugt werden. Weil dieses nur als Werkzeug während der Entwicklung dient, kann es rudimentär aufgebaut sein, sodass alle Einstellungen im Code selber erfolgen. Durch die erreichte Automatisierung der Erstellung eines Eingabedatensatzes können schnell durch geringfügige Veränderungen im Script verschiedene Permutationen erzeugt werden, die dem Testen des Frameworks dienen (siehe Abschnitt 4.7).

Um einen Optimierungsdatensatz zu erstellen, der die Entfernung eines Datenpunktes zur Modellaußenhülle eines 3D-Modells beinhaltet, wird ein Algorithmus entwickelt. Dieser berechnet die minimale Distanz eines Punktes zu einer tesselierten Oberfläche im 3D-Raum (siehe Anhang B.2) und wird als Funktion in `GridMaker` eingebunden.

5.4. Optimierungsvorschriften und Hilfsfunktionen

Wie in Abschnitt 3.2.4 dargestellt, können Optimierungsvorschriften und Hilfsfunktionen ähnlich aufgebaut werden. Beide Komponenten werden als Klassen umgesetzt. Allerdings ist die Erzeugung von Instanzen nicht nötig, da sie lediglich beinhaltete Daten kapseln, wie auch dem Framework zusätzliche Funktionalität liefern sollen. Daher werden Class Methods[7] verwendet, welche in Python anders als Static Methods auf Klassenattribute zugreifen können. Somit ist es möglich, dass eine Methode dem aufrufenden Element als Klassenattribute gespeicherte Instruktionen übergeben kann (siehe Abschnitt 4.2.1).

[7] https://www.python.org/dev/peps/pep-0252/#static-methods-and-class-methods

Dem Programmierer, der das Framework einsetzt, werden Basis-Klassen `Rule` und `Calculator` zur Verfügung gestellt, die Struktur, Attribute und Methoden von Optimierungsvorschriften und Hilfsfunktionen bereits vorgeben. Durch Vererbung kann er eigene Implementierungen erzeugen.

Wie in Abb. 5.2 zu sehen ist, können beide Komponenten Daten von Körpern durch `get_resources_cell()` anfordern, aber nur die Optimierungsvorschriften auch vom Optimierungsdatensatz mit `get_resources_grid()`. Dies hat den Grund, dass im Optimierungsdatensatz bereits alle Daten in der endgültig zu verarbeitenden Form vorliegen können, während Daten der Körper durch Modifikationen immer wieder verändert werden. Ein weiterer Unterschied besteht darin, dass eine Hilfsfunktion die berechnete Eigenschaft mit einem Namen bzw. Schlüssel versieht, welcher über `get_prop()` abgefragt werden kann. Die Optimierungsvorschrift kann ferner über `get_orientation()` die Ausrichtung der Teilungsebene zum Eigenschaftsgradienten vorgeben. Da die Methoden einfache Getter-Methoden sind und auf entsprechend benannte Attribute zugreifen, können sie in den Basis-Klassen implementiert werden.

Um die Optimierungsvorschrift bzw. die Eigenschaft zu berechnen, besitzen die Klassen die Class Method `apply()` bzw. `calc()`, welche als Argumente Daten aus Optimierungsdatensatz und Körper bzw. aus einem Körper annehmen. Diese unterscheiden sich zwischen den einzelnen Klassen und werden daher in den vom Programmierer geschriebenen Child-Klassen implementiert.

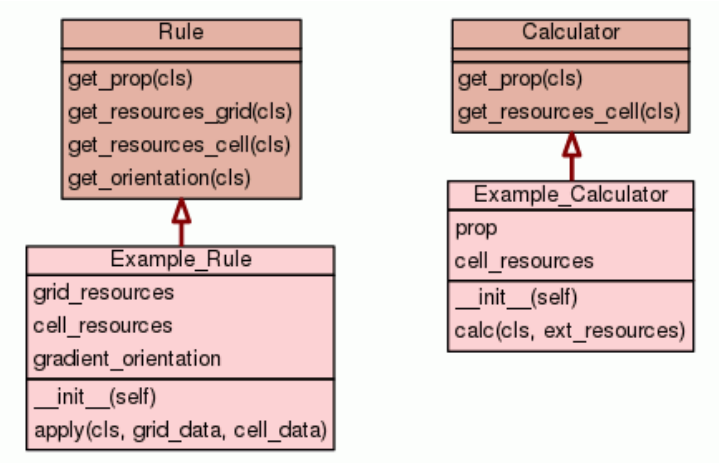


Abbildung 5.2.: Beispielhaftes Klassendiagramm einer Optimierungsvorschrift und einer Hilfsfunktion

5.5. Ausgabe und Tests

Zur Verifizierung der Framework-Implementierung werden Tests vollzogen. In der Umsetzungsphase werden Komponenten des Frameworks durch Unit-Tests auf ihre korrekte Implementierung geprüft. Durch Testfunktionen, die jeweils für eine Klasse zuständig sind, werden nötige Ressourcen gekapselt.

5.5.1. Ausgabe-Tests

Für in Abschnitt 4.7 beschriebene Testszenarien werden Ausgaben erzeugt. Es werden vier in Abb. 5.3 Optimierungsdatensätze mit dem Tool `GridMaker` (siehe Anhang B.1) erstellt, welche alle Szenarien abdecken. Die diese beinhaltenden Eingabedatensätze sind wie folgt eingestellt:

- Dimensionen des Optimierungsdatensatzes: 100 x 100 x 100 mm
- Auflösung: Ein Datenpunkt pro 5 mm^3
- Anfängliche Dimensionen der Körper: 10 x 10 x 10 mm
- Minimale Dimensionen der Körper: 0,8 x 0,8 x 0,8 mm

Die gewählte Auflösung des Optimierungsdatensatzes stellt einen Kompromiss zwischen Berechnungszeit und Genauigkeit dar. Es werden 8000 Datenpunkte generiert. Das angenommene Material zur Ermittlung der Dichte ist ColorFabb PLA/PHA[8], ein gebräuchliches Filament für die Herstellung von Prototypen und Kunstobjekten.

Zusätzlich wird ein einfaches 3D-Modell konstruiert (siehe Anhang A), anhand dessen die Optimierung der Körpergröße abhängig von der Entfernung zur Modellaußenhülle überprüft werden kann. Der Begrenzungsquader des 3D-Modells besitzt, korrespondierend zu den Optimierungsdatensätzen, Kantenlängen von 100 mm. Somit kann eine genügend hohe mögliche Auflösung und Differenzierung der resultierenden Füllstruktur erreicht werden. Durch Details wie den Aussparungen sollen komplexe Ausprägungen der Füllstruktur provoziert werden, die über jene, die bei einem einfachen, konvexen Modell zu erwarten sind, hinausgehen. Das 3D-Modell wird als STL-Datei[9], welche im Bereich der Additiven Fertigung de facto das Standardformat ist[10], gespeichert.

[8] https://colorfabb.com/files/FKUR/TD_BIO-FLEX_V_135001_en.pdf
[9] https://fileinfo.com/extension/stl
[10] https://www.3dsystems.com/quickparts/learning-center/what-is-stl-file

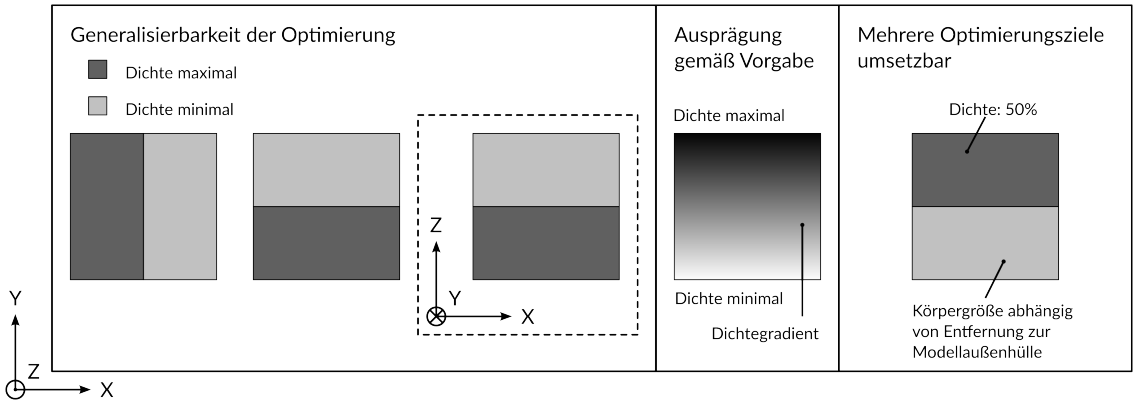


Abbildung 5.3.: Testszenarien für die Verifizierung des Frameworks, Darstellung in 2D-Aufsicht

Das Koordinatensystem einer SVG-Datei ist in mathematisch negativer Richtung definiert, sodass die Y-Achse gegenüber der einer STL-Datei invertiert ist. Slic3r führt die Koordinatentransformation beim Export in eine SVG-Datei nicht automatisch durch, weshalb eine in der Y-Achse gespiegelte Version des 3D-Modells zusätzlich für das Slicing erstellt werden muss. Das Slicing geschieht in Slic3r 1.2.9 mit 0,3 mm Schichtdicke.

5.5.2. Erstellung der Test-Eingabedatensätze

Mithilfe des Tools `GridMaker` werden vier Eingabedatensätze laut Abb. 5.3 erstellt:

- `testfile_gen1.json`
- `testfile_gen2.json`
- `testfile_gen3.json`
- `testfile_lim.json`
- `testfile_targets.json`

5.5.3. Erstellung der Ausgabedateien

Die erzeugten Eingabedatensätze werden in das Framework geladen und verarbeitet. Entstandene Füllstrukturen werden exportiert und anschließend durch den `Postprocessor`

(siehe Anhang B.1) in Schichten unterteilt. Dabei werden Schichten als einzelne SVG-Dateien gespeichert, damit die Betrachtung der Ergebnisse unabhängig von spezialisierter Software geschehen kann.

5.5.4. Auswertung

Für die Auswertung reicht eine Betrachtung weniger Schichten. Folgende Abschnitte zeigen, dass das Framework laut Vorgaben optimierte Füllstrukturen erzeugen kann. Die erreichte kleinste Körpergröße ist 1,25 x 1,25 x 1,25 mm und geht aus der progressiven Halbierung des Grundkörpers von 10 x 10 x 10 mm hervor. Die minimale Körpergröße von 0,8 x 0,8 x 0,8 mm wird also nicht erreicht.

Generalisierbarkeit und korrekte Ausrichtung

Für die Verifizierung der Generalisierbarkeit des Frameworks und der korrekten Ausrichtung der Füllstruktur wurden folgende Eingabedatensätze verarbeitet:

- `testfile_gen1.json`
- `testfile_gen2.json`
- `testfile_gen3.json`

Die Füllstrukturen (Abb. 5.4 und 5.5) wurden jeweils mit zwölf Iterationen des Optimierungsalgorithmus' erzeugt. Auswahlkriterium von Datenpunkten des Optimierungsdatensatzes war das lokale Minimum. Die Generalisierbarkeit des Frameworks sowie die korrekte Ausrichtung der entstandenen Füllstruktur wird bewiesen. Die Füllstruktur reagiert in allen Fällen wie erwartet auf die Vorgaben.

Einhalten von Vorgaben und Grenzbedingungen

Für die Verifizierung der Einhaltung von Vorgaben wurde der Eingabedatensatz `testfile_lim.json` verarbeitet. Die Füllstruktur (Abb. 5.6) wurde mit zwölf Iterationen des Optimierungsalgorithmus' erzeugt. Auswahlkriterium von Datenpunkten des Optimierungsdatensatzes war der lokale Medianwert.

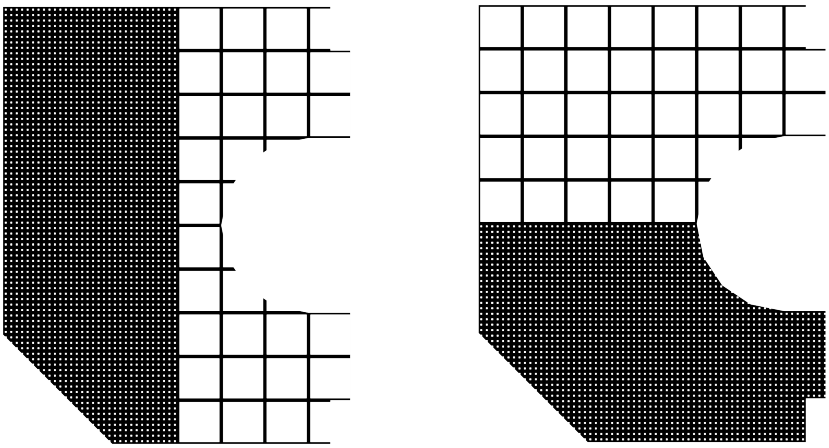


Abbildung 5.4.: Ausgabe der Füllstruktur mit minimaler bzw. maximaler Dichte, Vorgabe ausgerichtet an X-Achse (links) bzw. um 90° rotiert an Y-Achse (rechts), Schicht 98, z = 29,55 mm

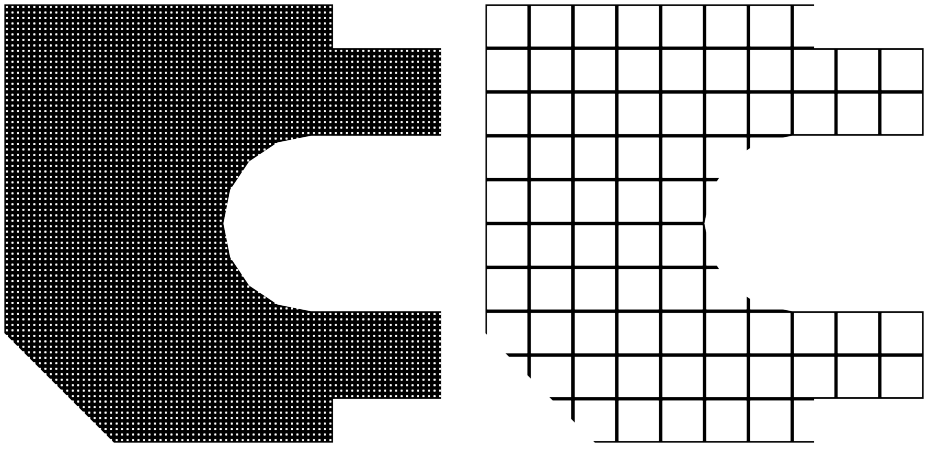


Abbildung 5.5.: Ausgabe der Füllstruktur mit minimaler bzw. maximaler Dichte, Vorgabe ausgerichtet an Z-Achse, Schicht 165, z = 49,65 mm (links) und Schicht 168, z = 50,55 mm (rechts)

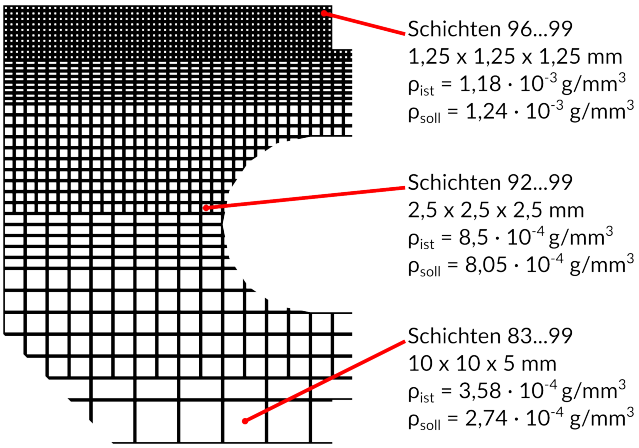


Abbildung 5.6.: Ausgabe der Füllstruktur mit Dichtegradienten als Optimierungsvorgabe, Schicht 98

Die Vorgabe der Dichte wird mit gewissen Abweichungen eingehalten, da Körper nur durch Halbierung modifiziert werden können. Die mininimale und maximale Körpergröße von 0,8 x 0,8 x 0,8 mm bzw. 10 x 10 x 10 mm wird eingehalten. Außerdem demonstrieren die rechteckigen Querschnitte, dass die Teilung von Köpern hauptsächlich orthogonal zum Eigenschaftsgradienten laut Vorgabe stattgefunden ist.

Umsetzung mehrerer Optimierungsziele

Für die Verifizierung der Umsetzung mehrerer Optimierungsziele wurde der Eingabedatensatz `testfile_targets.json` verarbeitet. Die Füllstruktur (Abb. 5.7) wurde mit sechs Iterationen des Optimierungsalgorithmus' erzeugt, um den Berechnungsaufwand bei der Erstellung von verschiedenen Versionen zu begrenzen. Auswahlkriterium von Datenpunkten des Optimierungsdatensatzes war das lokale Minimum für die Dichte. Das jeweilige Auswahlkriterium für den Abstand zur Modellaußenhülle ist Abb. 5.8 zu entnehmen.

Optimierungsziele werden jeweils erreicht und sind dabei räumlich scharf voneinander unterscheidbar. Wie Abb. 5.7 darstellt, nimmt die Körpergröße zur Modellaußenhülle hin ab, sodass diese durch eine dichtere Füllstruktur abgestützt werden kann. Dass die Füllstruktur nicht nur in zwei Dimensionen optimiert ist, kann anhand von Schicht 9 (Minimum) in Abb. 5.8 demonstriert werden. Diese befindet sich in der Nähe der unteren Modellaußenhülle, weshalb Körper stark unterteilt sind. Ebenso ist in Schicht 48 (Minimum) zu sehen, dass die Füllstruktur rechts unten auf den Überhang im Modell reagiert und so eine stärker ausgeprägte Zone mit kleinen Körpern bildet als an vertikalen Wänden, da die Distanz orthogonal zur Außenhülle gemessen wird.

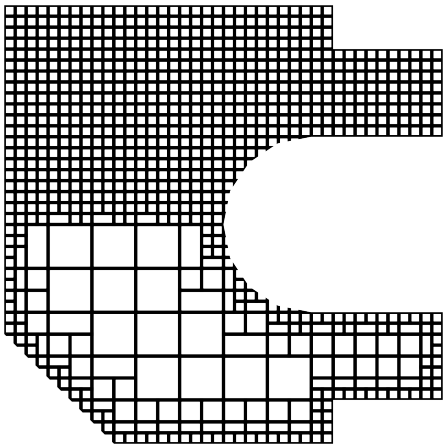


Abbildung 5.7.: Ausgabe der Füllstruktur mit zwei Optimierungsvorgaben, Schicht 298, z = 89,55 mm. Gut zu erkennen ist, dass das die Optimierung in der Nähe der Außenhülle zu kleinen Körpern führt.

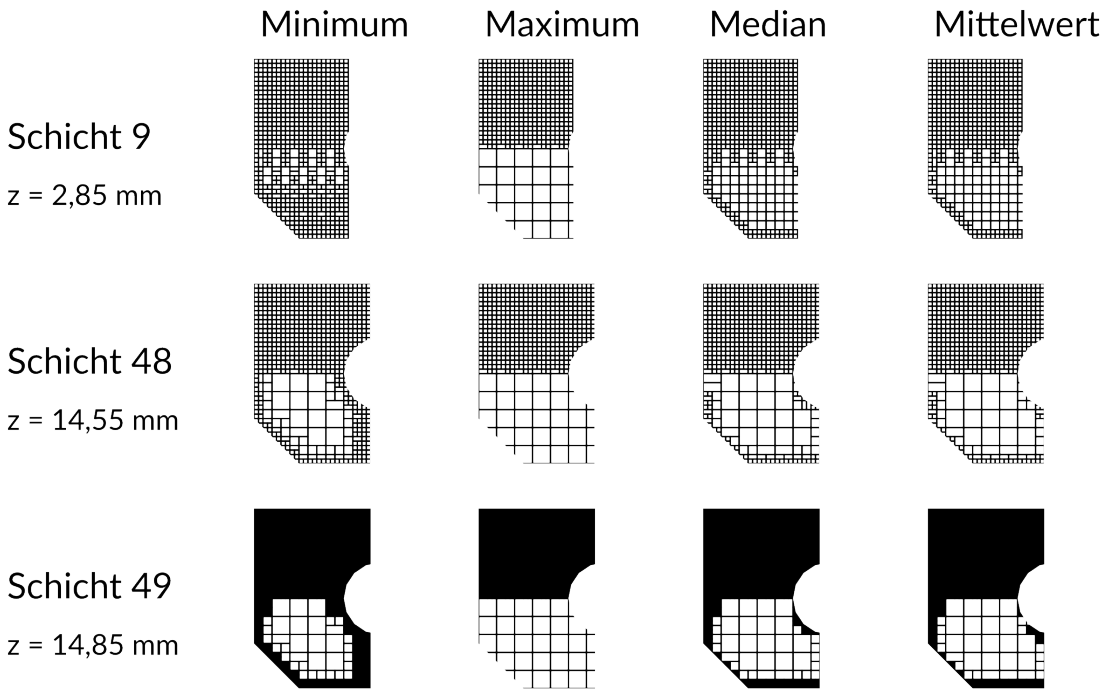


Abbildung 5.8.: Ausgabe der Füllstruktur in verschiedenen Ausprägungen. Schicht 49 weist Körperwände parallel zum Druckbett auf. Auswahlkriterien (Spalten) führen zu unterschiedlichen Mustern der Füllstruktur.

Weitere Erkenntnisse

Abb. 5.8 demonstriert zusätzlich, dass Variation des Auswahlkriteriums von Datenpunkten zu verschiedenen, teilweise fehlerhaften Ausprägungen (bis hin zu fehlender Optimierung bei Auswahl des Maximums) führen können. Daher kann festgehalten werden, dass die Wahl des Auswahlkriteriums ebenso gut überlegt und an die jeweilige, zu optimierende Eigenschaft angepasst sein muss, um eine erfolgreiche Optimierung der Füllstruktur durchzuführen.

Abb. 5.9 zeigt einen Vergleich zwischen Füllstrukturen, deren mögliche Isotropie eingeschränkt wurde bzw. deren Optimierung ohne Einschränkungen stattgefunden ist (siehe Abschnitt 4.4.3). Eine Bildung von scharfen Übergängen und dünnen, plattenartigen Strukturen, die vor allem in eine Hauptrichtung orientiert sind, lassen darauf schließen, dass die isotrope Füllstruktur nur eingeschränkt auf unerwartete Belastungen reagieren kann. Podrouzek u. a. (2019), Wu u. a. (2018) und Wu (2018) beschreiben ähnliche Problematiken, welche von Wu u. a. (2018) durch Einschränkung des lokalen Materialvolumens und Wu (2018) durch Filterung und Homogenisierung der Füllstruktur gelöst werden.

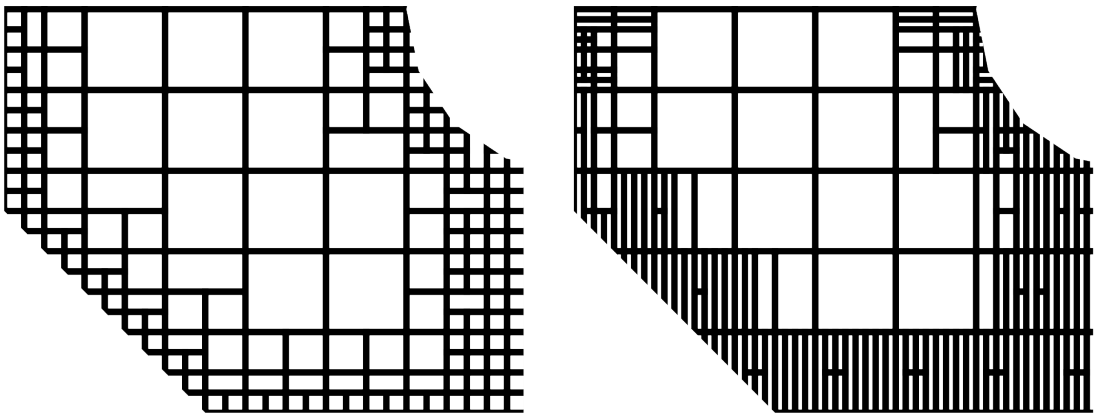


Abbildung 5.9.: Ausgabe des unteren Teils von Schicht 48 der Füllstruktur. Links: Isotropie eingeschränkt, Rechts: Isotropie zugelassen

Die Optimierung von Füllstrukturen ist in der Prototypenphase sehr rechenintensiv. Wie in Abschnitt 4.1.1 bereits erkannt wurde, steigt der Aufwand mit der Zahl der Körper. Somit steigt er ebenfalls mit der Anzahl der durchgeführten Iterationen der Optimierungsschleife, bis entweder die optimale Konfiguration oder die kleinste mögliche Körpergröße erreicht wird. Ein Vergleich anhand des Eingabedatensatzes `testfile_gen2.json` zeigt den Einfluss gewählter Iterationen und resultierender Anzahl von Körpern auf die Berechnungszeit

(siehe Tabelle 5.5.4). Ein effizienterer Algorithmus und Verbesserungen in der Programmstruktur wären in der Lage, diese zu senken. Zudem bietet es sich an, das Framework so anzupassen, dass es mehrere verfügbare Prozessoren zur Berechnung verwendet, da der Optimierungsalgorithmus aufgrund der Betrachtung einzelner Körper ohne deren räumlichen Kontext sehr gut parallelisierbar ist.

Tabelle 5.1.: Eckdaten bei verschiedener Anzahl von Iterationen des Optimierungsalgorithmus'

Iterationen	Anzahl erstellter Körper	minimale Körpergröße	Berechnungszeit
6	32500	2,5 x 2,5 x 2,5 mm	00:02:42
12	256500	1,25 x 1,25 x 1,25 mm	03:28:17

5.5.5. Versuchsweise Modifikation für Harz- und Pulververfahren

Da das Framework nur die Geometrie der Körper vorgibt, können die zum Druckbett parallelen Körperwände durch den `Postprocessor` bei der Schichtausgabe auf andere Verfahren als das FFF angepasst werden. In Abb. 5.10 sind beispielsweise versuchsweise (durch Einfügen weiterer Graphikelemente bei der Erstellung der SVG-Datei) Perforationen für Harz- und Pulververfahren implementiert, sodass unverbrauchtes Rohmaterial abfließen kann. Diese Modifikation verändert die letztendlichen Eigenschaften der Füllstruktur, was durch verändern der Optimierungsvorschriften berücksichtigt werden muss.

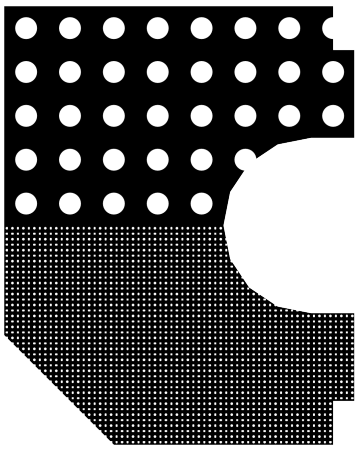


Abbildung 5.10.: Perforierte Wände der Füllstruktur, Schicht 99, z = 29,85 mm

6. Fazit

Ein nach Anforderungen funktionierender Framework-Prototyp wurde erstellt. Es kann gezeigt werden, dass dieser in der Lage ist, eine Füllstruktur für den 3D-Druck, insbesondere mit dem FFF-Verfahren, an verschiedene Vorgaben anzupassen und somit deren Eigenschaften zu verändern. Als Anpassungsmethode wurde die bereits von Wu u. a. (2016) und Wu (2018) erforschte Unterteilung einzelner Körper verwendet, die zu einer hohen lokalen Differenzierung führt. Die dadurch wachsende Anzahl an Körpern erweist sich wie erwartet als sehr rechenintensiv. Verbesserungen in der Software sind daher nötig, um eine effizientere Berechnung der Füllstruktur zu ermöglichen.

Als Innovation gegenüber bereits existierender Arbeiten (Rezaie u. a. (2013), Wu u. a. (2016), Wu u. a. (2017), Gopsill u. a. (2019), Wu (2018) und Groen u. a. (2019) verwenden extrudierte 2D-Körper) besitzt das Framework die Fähigkeit, dreidimensionale, zelluläre Strukturen zu erzeugen und diese lokal ebenfalls in drei Dimensionen auf mehrere Ziele hin zu optimieren. Allerdings kann das Framework strukturmechanische Zusammenhänge im Gegensatz vorangegangener Arbeiten nicht darstellen. Eine Optimierung auf mechanische Lasten ist also im aktuellen Zustand nur unter Zuhilfenahme weiterer Software möglich.

6.1. Ausblick

Die Erstellung der Füllstruktur mithilfe des entwickelten Frameworks stellt nur einen von vielen Arbeitsschritten zur Erzeugung einer Datei dar, die von einem 3D-Drucker in ein physisches Objekt umgesetzt werden kann. Nach der Prototypenphase soll das Framework weiter ausgebaut und optimiert werden, bevor es in einen Slicer oder eine größere 3D-Druck-Toolchain integriert werden kann. Insbesondere die Performance muss deutlich gesteigert werden, damit das Framework produktiv eingesetzt werden kann.

Die lokale Differenzierung der Füllstruktur (Abschnitt 5.5.4) wird momentan dadurch erreicht, dass mindestens ein Optimierungsziel, dessen Bedingungen getroffen werden, umgesetzt wird. Eine genaue Kontrolle über die Ausprägung der Füllstruktur kann somit nicht erreicht werden. Hierfür wäre die selektive (De-)Aktivierung von Optimierungszielen nötig, welches eine mögliche Erweiterung des Frameworks darstellt.

Ebenso ist der Übergang zwischen den Ausprägungen der lokalen Optimierungsziele scharf, d.h. Körper mit stark unterschiedlichen Dimensionen befinden sich in direkter Nachbarschaft. Durch eine Betrachtung der Nachbarschaft einzelner Körper können sanftere Übergänge realisiert werden, die zu einer höheren Robustheit der Füllstruktur führen, wie Wu (2018) beweist.

Eine modifizierte Ausrichtung der Körper kann Überbrückungen vermeiden, wie Wu u. a. (2016) mit extrudierten Rhomben demonstriert haben. Die dreidimensionale Natur der vom Framework eingesetzten Körper verlangt allerdings eine komplexe Transformation und einen gesteigerten Berechnungsaufwand bei der Erzeugung von Schichtinformationen.

6.2. Weitere Modifikationen der Füllstruktur

Wie bereits in Abschnitt 4.1.1 erwähnt, führt die progressive Unterteilung einzelner Körper dazu, dass die Füllstruktur lokal verschieden hoch aufgelöst sein kann. Die Form der gewählten Körper ist jedoch ein Kompromiss, um die Parkettierung des Modells zu vereinfachen. Durch die Kombination mit weiteren Anpassungstechniken, beispielsweise dem in Abb. 6.1 dargestellten Einsetzen von weiteren, an Eigenschaftsgradienten ausgerichteten Wänden innerhalb eines Körpers können Auflösung und Funktion der Füllstruktur stärker entkoppelt werden. Eine genauere Analyse der funktionsbestimmenden Elemente, wie von Garner u. a. (2018) vollzogen, ist hierfür nötig.

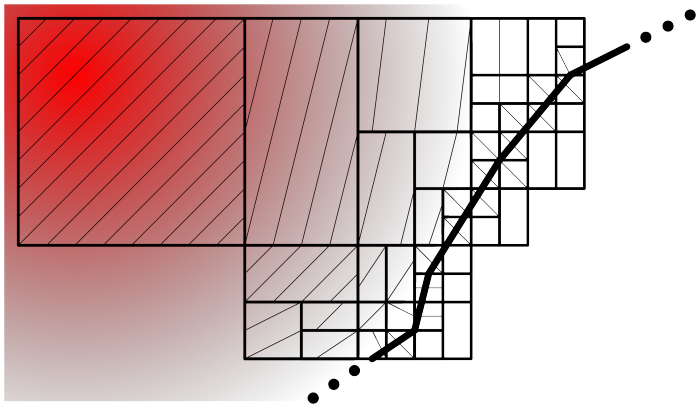


Abbildung 6.1.: Mögliche Erweiterung der Füllstruktur: Granularität (in Reaktion auf Abstand zur Modellaußenhülle - fett dargestellt) durch Teilung einstellbar, weitere Eigenschaften (auf Eigenschaftsgradient, rot, angepasst) durch Veränderung des Körpervolumens.

6.3. Veröffentlichung unter einer Open Source-Lizenz

Die Abstammung der aktuell erhältlichen 3D-Drucker, die das FFF-Verfahren verwenden, geht zu einem großen Teil auf das RepRap-Projekt zurück[1][2][3]. Durch die Verwendung von Open Source-Lizenzen konnte, wie von de Bruijn (2010) dokumentiert, eine große Basis von Entwicklern mobilisiert werden, die verschiedene Aspekte der Soft- und Hardware sowie Derivate davon geschaffen und verbessert haben. Das Potential der Open Source-Philosophie wird durch dieses[4] und weitere Beispiele an rascher Entwicklung von Hardware[5] und Software[6] aufgrund einer offenen Lizenzpolitik eindrucksvoll demonstiert. Daher wird auch das in dieser Arbeit entwickelte Framework mit einer Open Source-Lizenz versehen, um eine Entwicklung mit diesem und um dieses zu beschleunigen.

Die Mozilla Public License (MPL) 2.0[7] erlaubt es, die damit lizensierte Software in Projekten einzusetzen, die unter einer anderen Lizenz veröffentlicht werden. Damit ist sie weniger restriktiv als beispielsweise die GNU General Public License[8], die verlangt, dass derivative Werke komplett mit der gleichen Lizenz versehen werden. Überaus restriktive Free Software-Lizenzen würden den Einsatz des Frameworks in proprietärer Software behindern. Die MIT-Lizenz[9] und BSD-Lizenzen[10][11] erlauben dies zwar, führen aber zu dem Problem, dass die resultierende Software komplett unter einer proprietäreren Lizenz veröffentlicht werden kann. Der Vorteil der Mozilla Public License ist, dass unter ihr lizensierte Software mit Software kombiniert werden kann, die unter kompatiblen Lizenzen veröffentlicht worden sind. Proprietäre Derivate sind möglich, wobei jedoch der MPL-lizensierte quelloffen bleiben und eine Möglichkeit gegeben werden muss, diese zu beziehen[12]. Aufgrund der Flexibilität in der Anwendung für Open Source-Projekte wie auch für proprietäre Software wird das Framework unter der Mozilla Public License 2.0 veröffentlicht.

[1] https://www.reprap.org/wiki/About
[2] https://3dprintingindustry.com/news/dr-adrian-bowyer-receive-outstanding-contribution-3d-printing-award-107673/
[3] https://3dprint.com/192706/10-significant-things-results/
[4] https://reprap.org/wiki/Wealth_Without_Money
[5] https://enablingthefuture.org/about/
[6] https://www.oreilly.com/library/view/running-linux-third/156592469X/ch01s02.html
[7] https://www.mozilla.org/en-US/MPL/2.0/
[8] https://www.gnu.org/licenses/gpl-3.0.html
[9] https://opensource.org/licenses/MIT
[10] https://www.openbsd.org/policy.html
[11] https://opensource.org/licenses/BSD-3-Clause
[12] https://www.mozilla.org/en-US/MPL/2.0/FAQ/

Literaturverzeichnis

[Ben-Kiki u. a. 2009] BEN-KIKI, Oren ; EVANS, Clark ; NET, Ingy döt: *YAML Ain't Markup Language (YAML(TM)) Version 1.2*. yaml.org. 2009. – URL https://yaml.org/spec/1.2/spec.html

[Bray u. a. 2013] BRAY, Tim ; PAOLI, Jean ; SPERBERG-MCQUEEN, C. M. ; MALER, Eve ; YERGEAU, François ; COWAN, John: *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. World Wide Web Consortium. 2013. – URL https://www.w3.org/TR/REC-xml/

[de Bruijn 2010] BRUIJN, Erik de: On the viability of the open source development model for the design of physical objects - Lessons learned from the RepRap project. (2010). – ANR 23.99.45

[Clausen u. a. 2017] CLAUSEN, Anders ; ANDREASSEN, Erik ; SIGMUND, Ole: Topology optimization of 3D shell structures with porous infill. In: *Acta Mechanica Sinica/Lixue Xuebao* (2017), 06, S. 1–14. – DOI 10.1007/s10409-017-0679-2

[Crockford 2017] CROCKFORD, Douglas: *The JSON Data Interchange Syntax*. Ecma International. 2017. – ECMA-404

[Crump 1989] CRUMP, S. S.: *Apparatus and method for creating three-dimensional objects*. Google Patents. 1989. – US Patent 5121329A

[Deckard 1986] DECKARD, Carl R.: *Method and apparatus for producing parts by selective sintering*. Google Patents. 1986. – US Patent 4863538A

[Dietzfelbinger u. a. 2014] DIETZFELBINGER, Martin ; MEHLHORN, Kurt ; SANDERS, Peter: *Algorithmen und Datenstrukturen - Die Grundwerkzeuge*. Springer Vieweg, 2014. – ISBN 978-3-642-05472-3

[Furuhashi 2008] FURUHASHI, Sadayuki: *MessagePack specification*. 2008. – URL https://github.com/msgpack/msgpack/blob/master/spec.md. – Commit 3f0a9aa

[Garner u. a. 2018] GARNER, Eric ; KOLKEN, H.M.A. ; WANG, Charlie ; ZADPOOR, Amir ; WU, Jun: Compatibility in Microstructural Optimization for Additive Manufacturing. 26 (2018), 12, S. 65–75. – DOI 10.1016/j.addma.2018.12.007

[Gopsill u. a. 2019] GOPSILL, James A. ; SHINDLER, Jonathan ; HICKS, Ben J.: Using finite element analysis to influence the infill design of fused deposition modelled parts. In: *Progress in Additive Manufacturing* 3 (2019), S. 145–163. – DOI 10.1007/s40964-017-0034-y

[Groen u. a. 2019] GROEN, Jeroen ; WU, Jun ; SIGMUND, Ole: Homogenization-based stiffness optimization and projection of 2D coated structures with orthotropic infill. In: *Computer Methods in Applied Mechanics and Engineering* 349 (2019), 03. – DOI 10.1016/j.cma.2019.02.031

[Hull 1984] HULL, Charles W.: *Apparatus for production of three-dimensional objects by stereolithography*. Google Patents. 1984. – US Patent 4575330A

[Jones u. a. 2011] JONES, Rhys ; HAUFE, Patrick ; SELLS, Edward ; IRAVANI, Pejman ; OLLIVER, Vik ; PALMER, Chris ; BOWYER, Adrian: RepRap - the replicating rapid prototyper. In: *Robotica* 29 (2011), Nr. 1, S. 177–191. – DOI 10.1017/S026357471000069X

[Klein 2013] KLEIN, Bernd: *Leichtbau-Konstruktion - Berechnungsgrundlagen und Gestaltung*. Springer Vieweg, 2013. – ISBN 978-3-658-02272-3

[Knebl 2019] KNEBL, Helmut: *Algorithmen und Datenstrukturen - Grundlagen und probabilistische Methoden für den Entwurf und die Analyse*. Springer Vieweg, 2019. – ISBN 978-3-658-26512-0

[Lichtenegger 2009] LICHTENEGGER, Helga: Vorbild Natur. In: DEGISCHER, Hans P. (Hrsg.) ; LÜFTL, Sigrid (Hrsg.): *Leichtbau - Prinzipien, Werkstoffauswahl und Fertigungsverfahren*. Wiley-VCH Verlag, 2009, Kap. 1.1, S. 1–13. – ISBN 978-3-527-32372-2

[Liu u. a. 2018] LIU, Jikai ; GAYNOR, Andrew T. ; CHEN, Shikui ; KANG, Zhan ; SURESH, Krishnan ; TAKEZAWA, Akihiro ; LI, Lei ; KATO, Junji ; TANG, Jinyuan ; WANG, Charlie C. L. ; CHENG, Lin ; LIANG, Xuan ; TO, Albert. C.: Current and future trends in topology optimization for additive manufacturing. In: *Structural and Multidisciplinary Optimization* 57 (2018), Jun, Nr. 6, S. 2457–2483. – DOI 10.1007/s00158-018-1994-3. – ISSN 1615-1488

[Meagher 1982] MEAGHER, Donald: Geometric modeling using octree encoding. In: *Computer Graphics and Image Processing* 19 (1982), Nr. 2, S. 129–147. – DOI 10.1016/0146-664X(82)90104-6. – ISSN 0146-664X

[Ottmann und Widmayer 2017] OTTMANN, Thomas ; WIDMAYER, Peter: *Algorithmen und Datenstrukturen*. Springer Vieweg, 2017. – ISBN 978-3-662-55650-4

[Papula 2014] PAPULA, Lothar: *Mathematische Formelsammlung*. Springer Vieweg, 2014. – ISBN 978-3-8348-1913-0

[Podrouzek u. a. 2019] PODROUZEK, Jan ; MARCON, Marco ; NINČEVIĆ, Krešimir ; WANWENDNER, Roman: Bio-Inspired 3D Infill Patterns for Additive Manufacturing and Structural Applications. In: *Materials* 12 (2019), 02, S. 499. – DOI 10.3390/ma12030499

[Rammerstorfer und Daxner 2009] RAMMERSTORFER, Franz G. ; DAXNER, Thomas: Berechnungs- und Design-Konzepte für den Leichtbau. In: DEGISCHER, Hans P. (Hrsg.) ; LÜFTL, Sigrid (Hrsg.): *Leichtbau - Prinzipien, Werkstoffauswahl und Fertigungsverfahren*. Wiley-VCH Verlag, 2009, Kap. 1.2, S. 14–49. – ISBN 978-3-527-32372-2

[Rezaie u. a. 2013] REZAIE, R. ; BADROSSAMAY, M. ; GHAIE, A. ; MOOSAVI, H.: Topology optimization for fused deposition modeling process. In: *Procedia CIRP* 6 (2013), S. 521–526. – DOI 10.1016/j.procir.2013.03.098. – ISSN 2212-8271

[Shafranovich 2005] SHAFRANOVICH, Y.: *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Internet Engineering Task Force. 2005. – RFC 4180

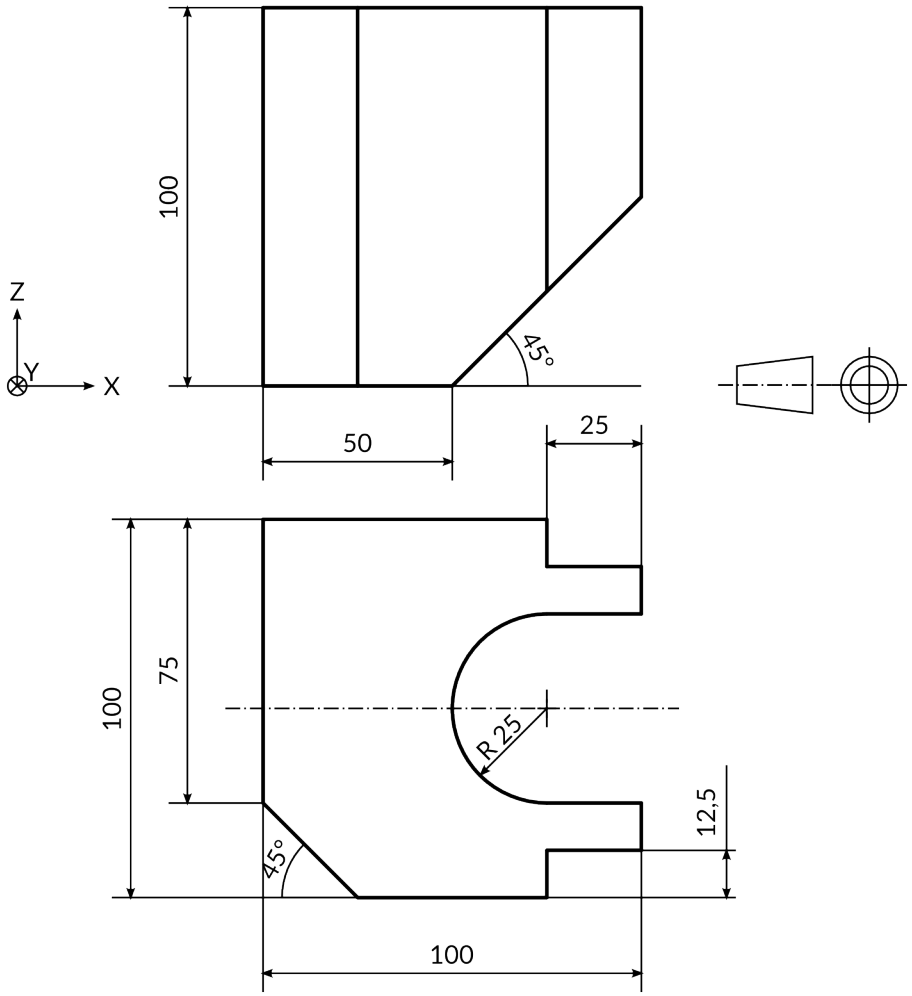
[Wu 2018] WU, Jun: Continuous Optimization of Adaptive Quadtree Structures. In: *Computer-Aided Design* 102 (2018), S. 72–82. – DOI 10.1016/j.cad.2018.04.008

[Wu u. a. 2018] WU, Jun ; AAGE, Niels ; WESTERMANN, Rüdiger ; SIGMUND, Ole: Infill Optimization for Additive Manufacturing – Approaching Bone-like Porous Structures. In: *IEEE Transactions on Visualization and Computer Graphics* 24 (2018), 02, S. 1127–1140. – DOI 10.1109/TVCG.2017.2655523

[Wu u. a. 2017] WU, Jun ; CLAUSEN, Anders ; SIGMUND, Ole: Minimum Compliance Topology Optimization of Shell-Infill Composites for Additive Manufacturing. In: *Computer Methods in Applied Mechanics and Engineering* 326 (2017), 08. – DOI 10.1016/j.cma.2017.08.018

[Wu u. a. 2016] WU, Jun ; WANG, Charlie ; ZHANG, Xiaoting ; WESTERMANN, Rüdiger: Self-Supporting Rhombic Infill Structures for Additive Manufacturing. In: *Computer-Aided Design* 80 (2016), 07. – DOI 10.1016/j.cad.2016.07.006

A. 3D-Modell für Tests am Framework



B. Inhalte der CD

B.1. Software

- `Cell Framework`
  Framework mit Testsoftware
  https://github.com/microGen/Cell_Framework
  Stand: commit 13e4d2e5a481c90606b3e1e65a6822fe12476d74
- `GridMaker`
  Software zur Erstellung von Eingabedatensätzen
- `Postprocessor`
  Software zum Export von Füllstruktur-Schichten in SVG-Dateien

B.2. Dokumente

- Dieses Dokument als PDF
- Anforderungsliste
- Tech Paper zur Bestimmung des minimalen Abstandes eines Punktes zur Oberfläche eines tesselierten 3D-Modells im 3D-Raum
- Ordner mit Readme-Dateien
- Ordner mit 3D-Modellen und zugehöriger Schichtausgabe für Tests am Framework

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Wichmann

Vorname: Nathaniel

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Entwicklung eines Software-Frameworks zur Erzeugung und Optimierung von Füllstrukturen für den 3D-Druck

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Bachelorarbeit ist erfolgt durch:

-

Hamburg	01.01.2020	
Ort	Datum	Unterschrift im Original