Masterarbeit

Adaptive Slicing-Algorithmen für low-cost 3D-Drucker mit mehreren Extrudern

Implementierung und Evaluation

Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Autor: Florens Wasserfall

(florens.wasserfall@informatik.uni-hamburg.de)

5876838

Gutachter: Prof. Dr. Peer Stelldinger

Dr. Norman Hendrich

Universität Hamburg, Departement Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2013/2014

Zusammenfassung

Beim Erzeugen von Objekten im *Fused Deposition Modelling* (FDM)-Verfahren muss immer ein Kompromiss zwischen Qualität und Druckgeschwindigkeit eingegangen werden. Um dieses Problem zu lösen, wurden bereits in den 1990er Jahren verschiedene Ansätze für adaptives Slicing entwickelt. Dabei wird die Schichtdicke an die lokale Oberfläche des Objekts angepasst, um nur in den Bereichen mit hoher Auflösung zu arbeiten, in denen es notwendig ist. Eine zweite, bislang wenig beachtete Möglichkeit ist das Variieren der Breite der erzeugten Kunststofffäden. Damit kann zusätzlich die horizontale Auflösung des Prozesses an die Objektstruktur angepasst und nur dort in hoher Qualität gedruckt werden, wo es notwendig ist.
In den letzten Jahren ist die Verbreitung kostengünstiger 3D-Drucker für den semiprofessionellen und Heimgebrauch stark angestiegen. Viele Hersteller und Projekte setzen dabei auf den Einsatz von Open Source Software, um die Modelldaten für den Druck aufzubereiten. Existierende Konzepte für adaptives Slicing wurden bislang in den verbreiteten Open Source Projekten gar nicht oder nur in Ansätzen umgesetzt. Der Einsatz mehrerer Extruder zur Verbesserung der Druckqualität wurde bislang wissenschaftlich nur wenig untersucht.

In dieser Arbeit werden bereits existierende Techniken für adaptives Slicing erweitert und im Rahmen eines bestehenden Softwareprojekts implementiert. Ein neuer Ansatz für adaptive Extrusion, sowohl mit einem als auch mit mehreren Extrudern, wird vorgestellt und ebenfalls implementiert. Für die Berechnung der Extrusionsbreite werden die Eigenschaften der Polygone analysiert, die den Umriss einer Schicht repräsentieren. Die abschließende empirische Evaluation der Algorithmen anhand einer Reihe von Testdrucken belegt eine signifikante Verbesserung der Qualität bei gleichzeitig reduzierter Druckzeit.

Abstract

Producing Objects with the *Fused Deposition Modelling* (FDM)-method requires operators to compromise between object quality and printing speed. A number of solutions has been presented during the last two decades, attempting to solve this problem by different approaches to adaptive slicing. The common procedure is to compute the layer thickness locally, based on the objects' surface characteristics. This only increases the resolution for regions with high requirements. Varying the extrusion width is a second possibility of taking the local object requirements into account. This potentially further increases both build quality and printing speed but has not been more deeply investigated yet.
In recent years, the development and availability of low-cost 3D-printers for semi professional and domestic use has increased dramatically. Many manufacturers and projects rely on one of the popular open source projects to prepare model data for printing. However, existing adaptive slicing approaches have not been implemented in those projects. The scientific investigation of the application of multiple extruders to improve printing quality has only just begun.

In this thesis, the implementation and enhancement of established adaptive slicing algorithms are presented, extending an existing software project. A new approach to adaptive extrusion for both single- and multi-extruder-systems is presented and implemented as well. The extrusion width is determined by analysing the characteristics of each layer's polygons. A significant improvement in quality and printing speed has been shown by an empiric evaluation of a number of actual printed objects.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Die Entwicklung von kostengünstigen 3D-Druckern für den semiprofessionellen bzw. privaten Einsatz hat in den letzten Jahren weltweit massiv zugenommen. Moilanen und Vadén [J. 12] stellen in ihrer Untersuchung ein starkes Wachstum seit etwa 2005/2006 fest und begründen dies mit dem Aufkommen des RepRap-Projekts [JHS+11]. De Brujin geht von einer exponentiellen Zunahme bei betriebenen RepRap Druckern aus [dB10], mit Wachstumsraten von mehreren 100% pro Jahr. In den letzten Jahren professionalisiert und kommerzialisiert sich diese Entwicklung. Eine Vielzahl von Herstellern bietet 3D-Drucker mit einer Preisspanne von einigen hundert bis wenigen tausend Euro bzw. Dollar an, auch im lokalen Einzelhandel sind erste Modelle verfügbar. Viele dieser Modelle verfügen bereits über zwei oder mehr Druckköpfe, mit dem primären Ziel, mehrfarbige Drucke und Stützstrukturen erzeugen zu können.

1.1. Problembeschreibung

Die meisten kostengünstigen 3D-Drucker werden entweder ohne, oder mit Open Source Software ausgeliefert und betrieben. Eine zentrale Aufgabe ist dabei das Slicing, also das Erzeugen von Maschinencode (G-Code) aus den CAD-Modellen. Aktuelle Open Source Slicer gehen dabei weitgehend statisch vor. Der Benutzer kann hauptsächlich eine Schichtdicke und die Druckgeschwindigkeit wählen, lokale Eigenschaften der verarbeiteten Objekte werden nicht berücksichtigt. Beispielsweise wird der aus einem einfachen Quader bestehende Sockel einer Figur mit derselben Auflösung wiedergegeben, wie die filigranen Strukturen des Kopfes. Dabei muss immer ein Kompromiss zwischen Qualität und Druckgeschwindigkeit eingegangen werden.
Bereits in den 1990er und 2000er Jahren wurde Forschung zu diesem Thema betrieben und Algorithmen für adaptives Slicing entwickelt. Der zentrale Ansatz ist dabei, die Schichtdicke nicht uniform für das gesamte Objekt zu verwenden, sondern abhängig von der Struktur der Oberfläche zu variieren. Die Ergebnisse dieser Arbeiten finden sich in heutiger Software nicht, oder nur in Ansätzen wieder.
Wenig beachtet wurde dabei die Qualität der Oberfläche. Häufig wurde vereinfachend davon ausgegangen, dass der extrudierte Kunststoff im Querschnitt einem eckigen Quader entspricht. Diese Vereinfachung berücksichtigt nicht die Tatsache, dass die Oberfläche gedruckter Objekte

mit zunehmender Schichtdicke sehr viel rauer wird.
Durch die Möglichkeit, mehrere Druckköpfe parallel zu betreiben, ergibt sich zusätzlich die Option, innerhalb eines Objekts den Düsendurchmesser stark zu variieren, um lokale feine Strukturen mit hoher Auflösung zu erzeugen, ohne den gesamten Druckvorgang stark zu verlangsamen.

1.2. Ziel der Arbeit

Ziel dieser Arbeit ist die Weiterentwicklung und Implementierung von adaptiven Slicing-Verfahren. Dazu sollen zum einen die bekannten Algorithmen zur Erzeugung variabler Schichtdicken geeignet implementiert und wenn nötig erweitert werden, um eine variable vertikale Auflösung zu erreichen. Zusätzlich soll eine Bewertung der Oberflächenqualität stattfinden und mit in den Entscheidungsprozess eingehen. Zum anderen soll untersucht werden, ob der Einsatz mehrerer Extruder für eine variable horizontale Auflösung geeignet ist. Dazu sollen Bewertungskriterien entwickelt werden, die es ermöglichen, die verfügbaren Extruder den lokalen Bereichen eines Objekts zuzuordnen.

Es soll in dieser Arbeit explizit um low-cost Open Source Hard- und Software gehen, damit die technischen Entwicklungen einem möglichst großen Nutzerkreis zugänglich sind. Dazu wird bereits existierende Hardware verwendet und gegebenenfalls angepasst und erweitert. Für die Implementierung der Algorithmen wird ein geeignetes Software-Projekt gewählt, um möglichst viel existierende Funktionalität nutzen zu können und die Ergebnisse für Dritte zugänglich zu machen.

Im letzten Schritt soll die Implementierung empirisch bewertet werden. Zentrale Fragen dabei sind:

- konnte die Qualität verbessert werden?
- Konnte die Druckzeit gesenkt werden?
- Ist der Einsatz mehrerer Extruder zur Beschleunigung des Prozesses geeignet?

1.3. Struktur der Arbeit

In Kapitel 2 werden kurz die Grundlagen von 3D-Modellen und der Zerlegung in 2.5D Schichten (Slicing) dargelegt. Anschließend wird ein Überblick über die Literatur zu den später verwendeten Verfahren gegeben. In Kapitel 3 wird die verwendete Hardware beschrieben und als Rahmen für die Implementierung infrage kommende Softwareprojekte vorgestellt. Es wird ein

Projekt als Grundlage für die Implementierung ausgewählt, die Auswahl begründet und ein Überblick über die interne Struktur der eingesetzten Software gegeben.

In Kapitel 4 wird die Implementierung der Kernalgorithmen für adaptive Schichtdicke und adaptive Extrusionsbreite beschrieben. Im ersten Teil werden die im Literaturüberblick dargelegten Verfahren umgesetzt und an einigen Stellen erweitert. Im zweiten Teil wird ein Verfahren zur Bestimmung einer geeigneten Extrusionsbreite und darauf basierend der Auswahl eines Extruders vorgestellt, das die Eigenschaften der Kontur einer Schicht analysiert.

Das 5. Kapitel befasst sich mit der Bewertung der Oberflächenqualität. Dazu wird diese empirisch analysiert und auf Basis eines geometrischen Modells der Zusammenhang zwischen Schichtdicke und Oberflächenqualität formalisiert. Anschließend wird ein Ansatz zur Integration der Fehlermaße für die Bewertung der Schichtdicken vorgestellt.

In Kapitel 6 wird die Effizienz der implementierten Algorithmen empirisch überprüft und bewertet. Anschließend wird im 7. Kapitel ein Fazit gezogen und ein Ausblick auf weitere mögliche Entwicklungen gegeben.

1.4. Definitionen und Konventionen

Für die Darstellung von Dezimalstellen in Zahlen wird die im englischen Sprachraum übliche Punkt-Notation verwendet, um die Konsistenz zum Quellcode und der verwendeten Literatur zu erhalten.

Im Text verwendete Fachbegriffe, technische Bezeichnungen und Namen werden wie folgt dargestellt:
- *Fachbegriffe* werden kursiv gedruckt
- Bezeichnungen und Auszüge aus Programmcode werden in `dicktengleicher` Schrift gesetzt.
- **Eigennamen** der Programme werden fetter gesetzt, um Verwechslungen zu vermeiden. Insbesondere der Name **Slic3r** wird damit deutlich von dem Prozess des Slicing abgegrenzt.

Für eine Reihe von Fachbegriffen werden sowohl in der Literatur, als auch im praktischen Umgang, die englischen Vokabeln verwendet. Es gibt nicht immer eine eindeutige Übersetzung. „Slicing“ etwa lässt sich übersetzen als „zerschneiden“ oder „in Scheiben schneiden“. Beides beschreibt zwar prinzipiell korrekt die technische Tätigkeit des Erzeugens von „Scheiben“ aus

einem Modell, wird aber im deutschen Sprachgebrauch nicht dafür verwendet. Für die folgenden Fachbegriffe werden deshalb (auch) die englischen Vokabeln verwendet.

Slicing bezeichnet im eigentlichen Sinne den Prozess, aus einem dreidimensionalen Objekt zweidimensionale Schichten zu generieren. Dazu wird der Schnitt einer Menge von Ebenen und der Modelloberfläche errechnet [PRD03c, S.1]. Bei Druckverfahren, in denen jede Schicht homogen durch Verkleben, Verschmelzen oder Aushärten erzeugt wird (Multi Jet Modelling (MJM), Selective Laser Sintering (SLS), Stereolithography (SLA)), benötigt der Drucker lediglich den Umriss der jeweiligen Schicht, etwa als Polygon. Die Fläche des errechneten Schnitts kann vollständig homogen bearbeitet werden, indem sie, ähnlich zur Ansteuerung eines Displays, punktweise „abgetastet“ wird. Beim Fused Deposition Modelling (FDM)-Verfahren fährt der Druckkopf die Kontur möglichst unterbrechungsfrei ab. Daher muss zusätzlich ein, nach verschiedenen Kriterien optimierter, Pfad errechnet werden, der die Kontur vollständig ausfüllt. Da sich beide Prozessschritte gegenseitig beeinflussen, wird im Folgenden der gesamte Prozess der Generierung von zweidimensionalen Schichten und der Verfahrwege als „Slicing“ bezeichnet.

Layer Durch den Slicing-Prozess erzeugte, einzelne Schicht des aufzubauenden Objekts. Die Begriffe „Layer“ und „Schicht“ werden in dieser Arbeit je nach Kontext synonym verwendet.

Extrusionsbreite [extrusion width] ist die tatsächliche Breite des extrudierten Kunststofffadens. Diese kann größer oder kleiner sein als der eigentliche Durchmesser der Düse, wenn das Volumen des extrudierten Materials größer oder kleiner ist als für den zurückgelegten Weg eigentlich benötigt.

Facet Ein Dreieck der Modelloberfläche, definiert durch 3 Punkte und die Oberflächennormale.

Extruder Ein Extruder besteht aus zwei Teilen: dem Vortriebsmechanismus und einer kontrolliert beheizten Düse. Diese können in einem Bauteil integriert, oder durch eine Bowdenhülle voneinander getrennt sein, um die schwere Vortriebsmechanik außerhalb der verfahrenden Achsen montieren zu können.

Filament Kunststoffdraht, der in der Düse des Extruders aufgeschmolzen wird. Filament ist das Rohmaterial, aus dem die Objekte synthetisiert werden.

Perimeter Der Außenbereich eines Objekts. Die Kontur jeder Schicht wird mehrfach vom Extruder abgefahren. Es wird sowohl jede geschlossene Umrundung als Perimeter bezeichnet als auch die Summe aller Umrundungen.

Infill Der Innenbereich eines Objekts. Dieser wird mit einem generischen Muster gefüllt, das ungefüllte Zwischenräume enthält, um Material, Druckzeit und Gewicht zu sparen.

Microlayering Ist die Schichtdicke im Perimeterbereich des Objekts gering, werden wenn möglich, mehrere Infill-Ebenen zu einer einzigen dickeren Ebene zusammengefasst, um den Prozess zu beschleunigen.

Support Stützstruktur, um überhängende oder frei schwebende Strukturen erzeugen zu können. Diese kann entweder aus dem gleichen Material bestehen, wie das Objekt und anschließend mechanisch entfernt werden oder aus löslichem Material, dass nach dem Druck in einer Flüssigkeit aufgelöst wird.

Raft Da es häufig zu Problemen mit dem Auftragen der ersten Schicht auf dem Druckbett kommt, kann zunächst eine grobe Struktur gedruckt werden, die als stabiler Grund für den eigentlichen Druckprozess dient und später entfernt wird.

Retraction Da der Kunststoff im Extruder unter Druck verarbeitet wird, läuft die Düse nach, wenn der Extrusionsprozess gestoppt wird. Um dies zu verhindern, wird das Filament etwas aus der Düse zurückgezogen.

2. Literaturüberblick

Das in den meisten aktuellen low-cost Druckern angewendete FDM-Verfahren wurde 1988 von Scott Cramp entwickelt, 1992 zum Patent angemeldet und von der Firma Stratasys kommerziell vertrieben [CLL03]. Dabei wird Kunststoff thermisch verflüssigt und mittels einer Düse schichtweise aufgetragen. Nach dem Abkühlen entsteht ein festes Objekt. Obwohl Alternativen existieren, verarbeiten die meisten Systeme hauptsächlich Modelldaten auf der Basis von Dreiecksfacetten (tesselation). Die gängigen Dateiformate sind *.stl*, *.obj* und *.amf*. Ein solches Modell wird im Englischen auch als *mesh*-Objekt oder *mesh* bezeichnet.

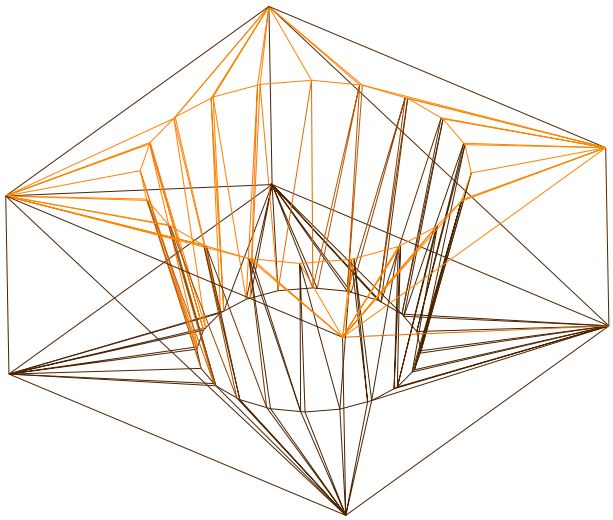
2.1. Stl-Dateien

Das Surface Tesselation Language oder Standard Triangulation Language (STL) Format wurde 1989 von der Firma 3D Systems, Inc. eingeführt [Bur89]. Es beschreibt die Oberfläche eines 3D-Modells durch eine Menge ungeordneter Dreiecke (*Facets*), jeweils bestehend aus drei Punkten und der Oberflächennormalen. Die Repräsentation der Daten ist stark redundant und enthält keine zusätzlichen Topologieinformationen, wie Nachbarschaftsbeziehungen zwischen Facets oder Punkten. Jeder Punkt ist Teil von mindestens drei benachbarten Flächen und wird daher mindestens dreimal gespeichert. Es existieren keine Informationen über Nachbarschaftsbeziehungen oder sonstige Ordnungen der Flächen. Ein Beispiel einer *.stl*-Datei ist in Abb. 2.1 gegeben.

Die Dateiformate *.obj* und *.amf* enthalten mehr Informationen als *.stl*-Dateien. Sie sind aber grundlegend gleich strukturiert und werden im Wesentlichen mit den selben Algorithmen verarbeitet. Dazu werden alle Eingaben zunächst in ein programmspezifisches, internes Format überführt. Daher werden im Folgenden, wenn nicht explizit anders erwähnt, *.stl*-Dateien und ihre Verarbeitung beschrieben.

2.2. Slicing

Beim Slicen wird das dreidimensionale Modell in 2.5-dimensionale Scheiben zerlegt, indem jeweils der Schnitt einer horizontalen Ebene mit allen Flächen des Modells berechnet wird. Das



```
1  solid Example_Model
2    facet normal -1 0 0
3      outer loop
4        vertex 0 0 0
5        vertex 0 0 10
6        vertex 0 20 0
7      endloop
8    endfacet
9    facet normal -1 0 0
10     outer loop
11       vertex 0 20 0
12       vertex 0 0 10
13       vertex 0 20 10
14     endloop
15   endfacet
16   ...
```

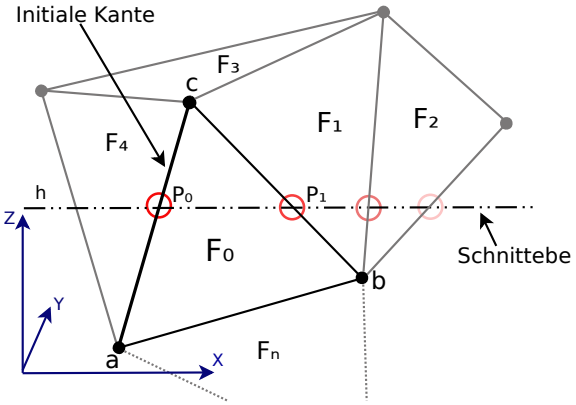
Abbildung 2.1. – Beispiel einer .stl-Datei. **Links**: grafische Darstellung als Gittermodell. **Rechts**: Auszug aus der in ASCII codierte Datei.

Ergebnis dieser Operation ist eine Menge von gerichteten Linien, aus denen im nächsten Schritt ein Polygon erzeugt wird. Per Konvention ist festgelegt, dass externe Konturen gegen den Uhrzeigersinn (CCW) und interne Konturen mit dem Uhrzeigersinn (CW) gerichtet sind [Tyb98, S.11]. Die zu füllende Fläche liegt also immer auf der linken Seite der Polygonlinie. Da die Dreiecke in *.stl*-Dateien unsortiert vorliegen, müsste die aus dem Schnitt einer Ebene mit dem Objekt resultierende Menge an Linien sortiert werden, um ein zusammenhängendes Polygon zu erzeugen. Um diesen Prozess zu optimieren, haben Rock und Wozny den „marching algorithmus“ entwickelt [RW91]. Dabei wird initial für jedes Dreieck festgestellt, mit welchen drei Dreiecken es direkt benachbart ist. Um den Schnitt einer Ebene zu berechnen, wird zunächst ein beliebiges Dreieck gesucht, zwischen dessen Minimum und Maximum die Schnittebene liegt (F_0 in Abb. 2.2a). Der Schnittpunkt P zwischen Ebene und einer Kante des Dreiecks wird berechnet mit:

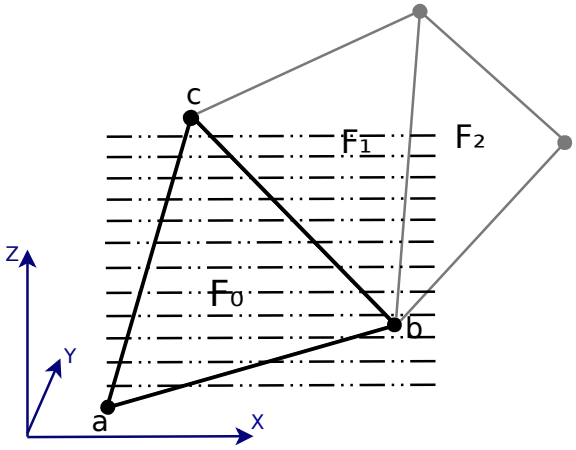
$$P_0 = a + (a - b) \cdot \frac{h - b_z}{a_z - b_z} \tag{2.1}$$

Wobei a ($[a_x, a_y, b_z]$) und b die Endpunkte der Kante und h die Höhe der Schnittebene sind. Da $b_z < h$ wird im 2. Schritt der Schnitt mit $\overline{bc}$ berechnet. Die Nachbarschaftsbeziehung zwischen F_0 und F_1 ist bereits bekannt und der Prozess kann in F_1 analog fortgesetzt werden. Der Polygonzug ist geschlossen sobald die initiale Kante erreicht wird. Da es mehrere geschlossene Polygonzüge in einer Ebene geben kann, muss für jedes Dreieck eines Objekts geprüft werden ob es von der Ebene geschnitten wird. Der Algorithmus wird für jeden Polygonzug ausgeführt.

Wenn die Positionen aller Schnittebenen im Voraus bekannt sind und nicht vom Ergebnis der aktuellen Berechnung abhängen, lässt sich das Verfahren parallelisieren, indem n Schnittebenen gleichzeitig berechnet werden (Abb. 2.2b).



(a) Berechnung der Schnittpunkte zwischen Ebene und den Kanten der Dreiecke, farblich in der Reihenfolge hervorgehoben die der Algorithmus durchläuft.



(b) Berechnung der Schnittlinien aller Ebenen für ein Dreieck.

Abbildung 2.2.

Aus den Polygonen wird *G-Code* erzeugt, den der 3D-Drucker ausführen kann. Dazu wird zunächst durch *Offsetting* des Polygons eine Anzahl von *Perimetern* erzeugt. Diese Linien werden vom Drucker abgefahren. Der verbleibende Teil (*Infill*), wird mit einem regelmäßigen Muster gefüllt.

2.3. Verbesserungsansätze

Ein fundamentales Problem aller Layered-Manufacturing-Verfahren ist der Kompromiss zwischen Verabeitungsgeschwindigkeit und Qualität. Je größer Schichtdicke und Extruderbreite sind desto geringer ist i.A. die Druckzeit. Dabei lässt sich unterscheiden zwischen der Qualität der Oberfläche und der Abweichung des gedruckten Objekts vom Modell. Für FDM-Verfahren existieren bereits seit den 90/2000er Jahren diverse Ansätze zur Verbesserung. Diese Arbeiten fokussieren sich darauf, eine möglichst hohe vertikale Auflösung bei geringer Druckzeit zu erreichen. Sie werden klassischerweise mit „Adaptive Slicing“ bezeichnet. Erst in den letzten Jahren gibt es Ansätze, auch die horizontale Auflösung durch Veränderung der Extruderparameter zu verbessern.

2.3.1. Adaptives Slicing

Durch das Synthetisieren eines Objekts aus diskreten Schichten ergeben sich einige fundamentale Probleme:

Staircase/Stairstepping Oberflächen, die nicht exakt horizontal oder vertikal verlaufen, können nur approximiert werden. Dabei wird die Ungenauigkeit größer, je dicker die Schichthöhe gewählt wurde. Da das Erzeugen einer dünnen Schicht vergleichbar viel Zeit erfordert, wie das Erzeugen einer dicken Schicht, sind Drucke mit höherer Qualität zeitintensiver.

Verzerrung Durch den Stairstepping-Effekt gewinnt oder verliert jede geneigte Oberfläche eines Objekts an Volumen, da die einzelnen Stufen innerhalb oder außerhalb der Solloberfläche liegen müssen (Abb. 2.3). Dieser Effekt wird als „Volumetric Error“ [TFBA98] oder „Containment Problem“ [PRD03c] bezeichnet. Hinzu kommt, dass nur diskrete Höhenstufen erreicht werden können. Dadurch ergeben sich im Allgemeinen Ungenauigkeiten in der Z-Dimension des gedruckten Objekts an horizontalen Flächen.

Oberflächenwelligkeit In den meisten Arbeiten wird der Extruderfluss optimistisch als Rechteck approximiert. In der Praxis ergibt sich durch das Auftragen von elliptischen Extrudaten auch bei exakt vertikalen Flächen eine gewellte Oberfläche.

Als Lösungsansatz für einige dieser Probleme werden im folgenden Abschnitt adaptive Slicing-Verfahren vorgestellt. Abgesehen von den auftretenden Ungenauigkeiten, ist der Druckprozess besser automatisierbar, wenn der Nutzer keine Entscheidungen über die Schichtdicke treffen muss, weil diese algorithmisch bestimmt wird.

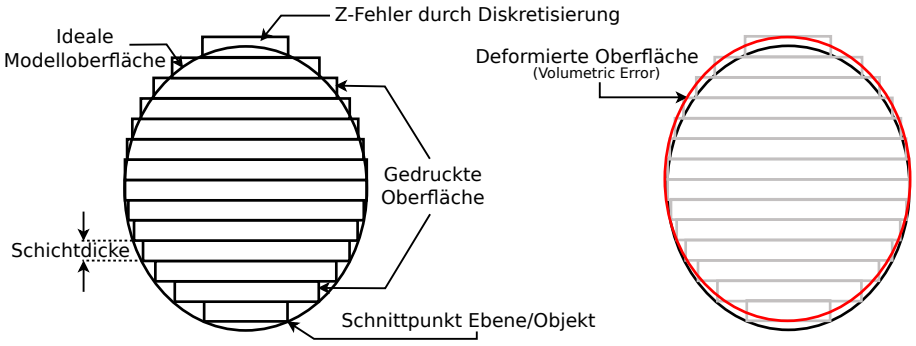


Abbildung 2.3. – Der Stairstepping-Effekt und seine Folgen. **Links:** Gedruckte Oberflächen approximieren das Modell in schrägen Regionen stufenförmig. Abhängig davon, auf welcher Höhe einer Schicht der Slicer den Schnitt berechnet und der Neigungsrichtung der Modelloberfläche, entsteht eine positive (oben) oder negative (unten) Abweichung von der gewünschten Kontur. **Rechts:** Resultierende Verformung des Objekts.

Cusp Height

Um gleichzeitig den Stairstepping-Effekt und die nötige Druckzeit zu minimieren wurden eine Reihe von Verfahren vorgestellt. Dolenc und Mäkelä [DM94] führten bereits 1994 das häufig

verwendete Fehlermaß *cusp height* ein, um ausgehend von der aktuellen Schicht die optimale Dicke der nächsten Schicht zu errechnen.

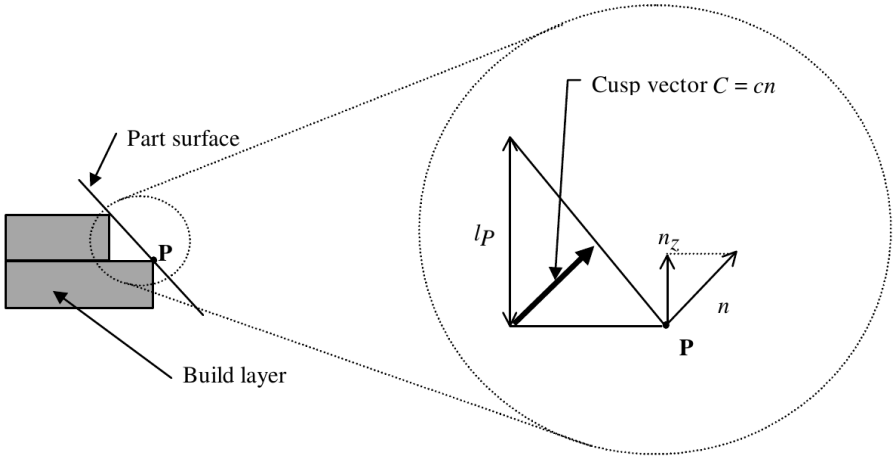


Abbildung 2.4. – Berechnen der optimalen Schichthöhe aufgrund des Cusp-Vektors und der Oberflächennormalen des Objekts [DM94].

Dazu wird für eine Menge von Punkten P auf der aktuellen Kontur, unter Verwendung des vorgegebenen maximalen Fehlers C_{max}, die Höhe der nächsten Schicht c für die Umgebung des jeweiligen Punktes ausgerechnet. Die Schichthöhe ist nach oben durch die technisch maximale Höhe L_{max} begrenzt. Für einen Punkt lautet die Gleichung:

$$c = min\{L_{max}, C_{max}/n_z\} \tag{2.2}$$

wobei n_z die Z-Komponente der Oberflächennormale am Punkt P ist.

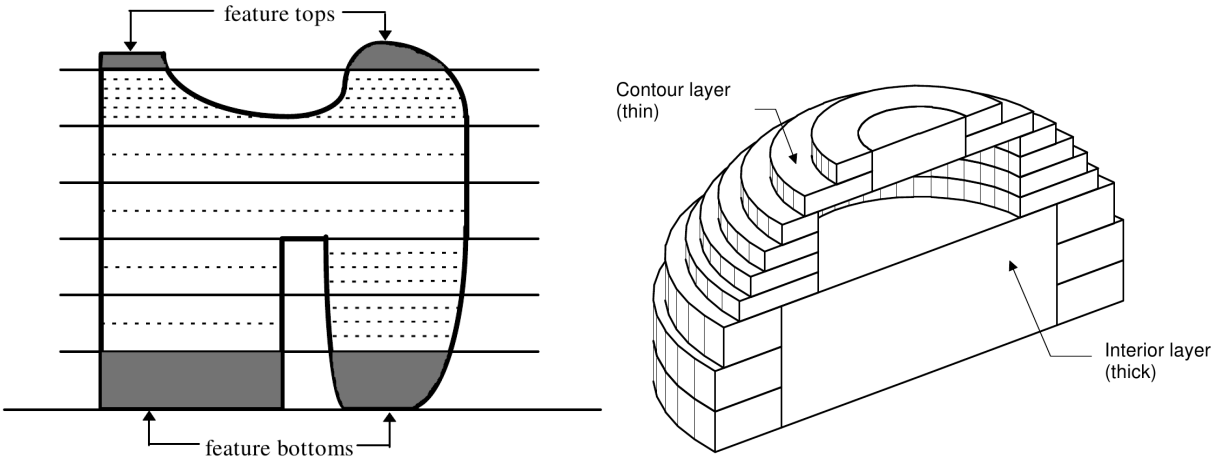
Local Adaptive Slicing

Tyberg [Tyb98] erweitert das Verfahren um die Möglichkeit bei Verzweigungen oder mehreren Objekten jede Komponente unabhängig adaptiv zu slicen, *local adaptive slicing*. Dies verhindert, dass ein Teil des Objekts die Verwendung kleiner Schichthöhen für das gesamte Objekt erzwingt. Tyberg unterteilt das Objekt dazu in dickere Scheiben, *Slabs*, stellt die Beziehung zwischen mehreren Konturen in zwei Ebenen her und wendet anschließend adaptives Slicing nach Dolenc [DM94] für jeden gefundenen *Slab* an, siehe Abb. 2.5a. Dies ermöglicht auch Parallelisierung.
Bei diesem Verfahren wird davon ausgegangen, dass mehrere Objekte technisch wie ein einzelnes Objekt behandelt werden, welches aus mehreren disjunkten Komponenten besteht. Die Zuordnung der einzelnen Umrisse einer Schicht zu denen der nächsten Schicht erfolgt geome-

trisch, dazu wird die Entfernung und relative Orientierung der Polygone zueinander ausgewertet.

Interiour/Exteriour

Sabourin [Sab96] verwendet ebenfalls das Konzept von dickeren Slabs, unterteilt aber zusätzlich jeden Slab in einen Innen- und einen Außenbereich, siehe Abb. 2.5b. Im Innenbereich kann immer die maximale Schichtdicke verwendet werden, und außen erfolgt adaptives Slicing nach Dolenc. Dazu wird im Kern der Contour Offsetting Algorithmus nach [FA94] verwendet. Das Verfahren beschleunigt den Prozess, da im Großteil des Objekts hohe Schichtdicken verwendet werden können. Sabourin betrachtet dabei anders als Tyberg, immer das gesamte Objekt, ohne Unterscheidung von Verzweigungen.



(a) Aufteilung des Objekts in Slabs und unabhängiges adaptives Slicing; [Tyb98].

(b) Aufteilung des Objekts in Innen- und Außenbereiche, anschließend im Außenbereich adaptives Slicing; [Sab96].

Abbildung 2.5. – Ansätze zur Verwendung unterschiedlicher Auflösungen in einem Objekt.

Oberflächenrauigkeit

Pandey et al. [PRD03a] haben untersucht wie sich die Qualität einer Oberfläche in Abhängigkeit ihres Neigungswinkels verhält. In ihrer Arbeit vermessen sie die Geometrie der gedruckten Oberflächen und approximieren diese als parabolische Funktion. Die gängige Approximation der Kontur durch Rechtecke wird durch das Modell signifikant verfeinert und eine zusätzliche Bewertungsmöglichkeit der Druckqualität geschaffen. Sie verwenden die mittlere Rauheit R_a

als Kennzahl zur Beschreibung der Oberfläche. Diese wird in Abhängigkeit von Schichtdicke und Oberflächenneigung, empirisch ermittelt, mit

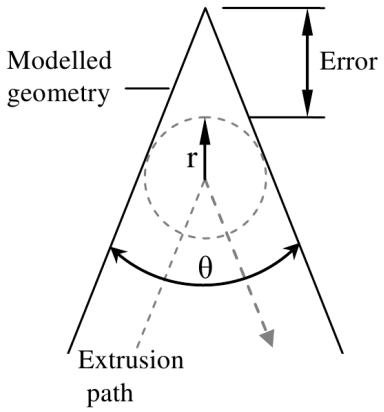
$$R_a(\mu m) = (69.28 - 72.36)\frac{t(mm)}{cos\,\theta} \tag{2.3}$$

angegeben, wobei t die Schichtdicke (*thickness*) und θ der Neigungswinkel der Oberfläche ist. In einer weitere Arbeit [PRD03b] verwenden sie diese Gleichung zur Steuerung der Schichtdicke als Alternative zur *cusp heigth*. Die Schichtdicke wird für ein vorgegebenes R_a berechnet mit:

$$t = \frac{R_a\,cos\,\theta}{70.82} \tag{2.4}$$

2.3.2. Adaptive Extrusion

Neben den durch die Diskretisierung des Modells in vertikaler Richtung induzierten Fehlern, ist bei FDM-Verfahren auch die horizontale Auflösung prinzipbedingten Effekten unterworfen. FDM-Düsen sind in der Regel achsensymmetrisch, also kreisförmig, damit sie nicht normal zum Extrusionspfad ausgerichtet werden müssen. Modelle mit Kurvenradien R kleiner als der Extruderradius r können damit nicht fehlerfrei abgebildet werden. Brooks et al. [BRA+] geben eine Abschätzung des Fehlers für spitze Winkel in Abhängigkeit vom Extruderradius (Abb. 2.6).



$$Error = \frac{r}{sin(\frac{\theta}{2})} - r \tag{2.5}$$

Abbildung 2.6. – Geometrischer Fehler beim drucken winkelförmiger Geometrien, aufgrund der runden Extruderdüse. Abbildung aus [BRA+]

In der Praxis ist der Fehler deutlich größer als in dieser Rechnung, da sich die Pfade beim Betreten und Verlassen des Winkelbereichs nicht überschneiden dürfen.
Wird nur ein statischer Extruder verwendet, steigt die Druckzeit eines Objekts linear mit abnehmendem Extruderdurchmesser:

$$Druckzeit = \frac{Volumen}{D \cdot f \cdot l} \tag{2.6}$$

mit $Volumen$ = Volumen des Objekts [mm], D = Durchmesser des Extruders [mm], f = Druckgeschwindigkeit [mm/s], l = Schichtdicke.

Brooks et al. benennen drei Optionen um die Druckzeit zu verringern:

- Die horizontale Verfahrgeschwindigkeit erhöhen.
- Das Volumen des im Objekts benötigten Materials verringern.
- Den Durchmesser der Düse vergrößern.

In ihrem Ansatz verwenden sie eine zweistufige Düse, um den Durchmesser je nach Region (Außen/Innen) anpassen zu können. Sie weisen explizit darauf hin, dass sich derselbe Effekt auch mit dem Einsatz von zwei statischen Düsen erzielen ließe.

3. Verwendete Grundlagen

Damit die Ergebnisse dieser Arbeit einem möglichst großen Nutzerkreis zugänglich sind, wird grundsätzlich offene (open source), und möglichst kostengünstige Hard- und Software verwendet. Es soll auf bereits existierenden Grundlagen aufgebaut werden, damit die Ergebnisse direkt verwendet werden können.

3.1. Hardware

Zur Evaluation der implementierten Algorithmen kommen zwei auf dem Reprap-Projekt [JHS+11] basierende FDM 3D-Drucker zum Einsatz. Ein weitgehend unmodifizierter Prusa Mendel I [2] mit nur einem direkt getriebenen Extruder und ein für den Einsatz von drei Extrudern modifizierter Prototyp. Da es problematisch ist, die Masse von drei Extrudern zu verfahren, sind nur die beheizten Düsen auf der Achse montiert, die Mechanik für den Vortrieb ist am Gehäuse befestigt und über einen PTFE-Schlauch mit der Düse verbunden (*Bowden-Extruder*). Die verschiedenen Extrudertypen weisen signifikante Unterschiede im Verhalten auf, da der Transport des Filaments über einen Schlauch eine gewisse Kompression zulässt.

3.1.1. Fähigkeiten / Einschränkungen der Extruder

Auf den Druckern sind je ein 0.5mm, 0.35mm und 0.25mm Bowdenextruder bzw. ein 0.5mm Extruder mit direktem Antrieb parallel installiert, um ein großes Extrusionsbreitenintervall abdecken zu können. Direkt getriebene Extruder neigen dazu, weniger zu tropfen und benötigen daher geringere *Retraction* bei Sprüngen.
Für den parallelen Betrieb mehrerer Extruder ist es entscheidend, das Filament weit zurückzuziehen, um das Tropfen auch über längere Zeit zu verhindern. Besonders bei kleinen Düsendurchmessern ist dies problematisch, weil die Extrusion nach einer längeren Phase starker Retraction länger benötigt, um sich wieder zu stabilisieren. Der Effekt wird in Abb. 3.1 sichtbar. In den unteren Schichten wurde abwechselnd der kleine Extruder (0.25mm, rot) für außen und der große Extruder (0.5mm schwarz) für innen verwendet. Die Initialisierung des großen Extruders nach dem Wechsel funktioniert signifikant besser.
Um trotzdem einen abwechselnden Betrieb der Extruder zu ermöglichen, werden sie bei einem Wechsel gereinigt, indem einige Millimeter Filament extrudiert und der Extruder anschließend

an einer Silikonkante abgestreift wird. Der Reinigungsprozess dauert ca. 30 Sekunden, dies kann eine signifikante Verzögerung des Drucks bedeuten.

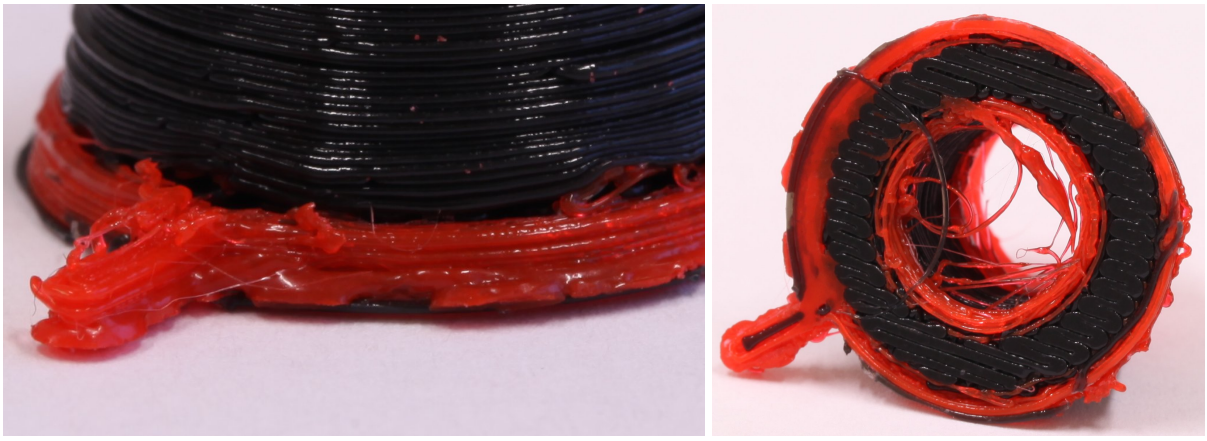


Abbildung 3.1. – Auswirkung unsauberer Retraction nach Extruderwechseln. Bei großem Düsendurchmesser (schwarz, 0.5mm) ist der Effekt deutlich schwächer als bei kleinem Durchmesser (rot, 0.25mm).

3.2. Software

Das Ziel dieser Arbeit ist weniger ein Proof-of-Concept, sondern die Implementierung generisch einsetzbarer Software. Dazu soll auf vorhandener Infrastruktur aufgebaut und ein existierendes Projekt erweitert werden. Im Folgenden werden kurz die wichtigsten verfügbaren Programme vorgestellt und bewertet. Die wesentlichen Kriterien dabei sind:

- Open-Source-Lizenz
- Aktive Entwicklung und Verankerung in der Community
- Generische Modellierung der Druckereigenschaften, insbesondere der Extruder und ihrer Parameter
- Unterstützung für variable Schichthöhen
- Horizontale Aufteilbarkeit des Objekts
- Lokale Parameter (z.B. Extruderbreite in jedem Layer / jedem Polygon) um lokal Variationen abbilden zu können

3.2.1. Slic3r

Slic3r [10] ist eines der momentan populärsten Open Source Slicing-Werkeuge. Es wurde ursprünglich in Perl geschrieben, performancekritische Teile wurden mittlerweise nach C/C++ portiert und per XS angebunden. Es steht unter der AGPLv3-Lizenz und wird von Alessandro

Ranellucci aktiv entwickelt.
Slic3r verfügt über eine optionale GUI, ist aber auch explizit für den Einsatz als Backend vorgesehen und wird von diversen Projekten verwendet.
Die Konfiguration der Maschinenparameter erfolgt modular. Extruder und Filament werden jeweils mit ihren Parametern als eigene Objekte modelliert und können untereinander kombiniert werden. Während des Slicing-Prozesses stehen damit Informationen über Anzahl der Extruder, Düsendurchmesser, eingelegtes Filament etc. zur Verfügung. Somit ist die Grundlage für eine algorithmische Entscheidung darüber, welcher Extruder mit welchen Parametern in einem Teilbereich eingesetzt wird, bereits vorhanden.
Es wird prinzipiell eine uniforme Schichtdicke für das gesamte Objekt verwendet, allerdings ist es manuell möglich, vertikalen Regionen unterschiedliche Schichtdicken zuzuweisen.

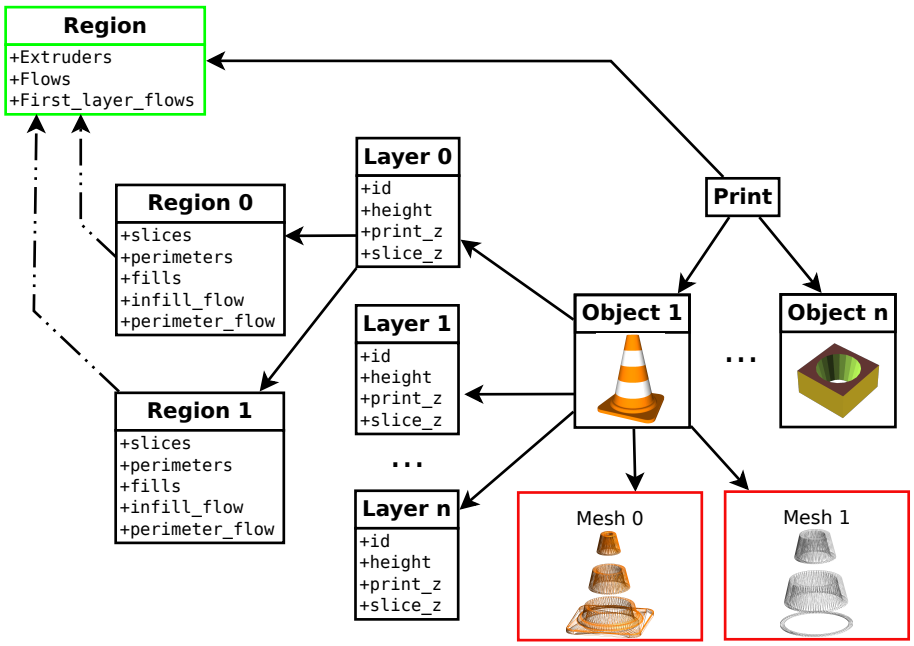


Abbildung 3.2. – Überblick über die Struktur der Daten und Parameter in **Slic3r**. Das `Print`-Objekt enthält die Daten für alle zu druckenden Objekte und die Druckerparameter. Aus den *Mesh*-Objekten[1] werden im Slicing-Schritt die `Layer`-Objekte erzeugt. Diese enthalten lokale Daten zu jedem Layer (Höhe, Dicke) und Referenzen auf die `Region`-Objekte. Jedes `Region`-Objekt entspricht einem Mesh (bei Multimaterialobjekten). In den `Regions` liegen die eigentlichen Polygone (slices) sowie Kopien des globalen (grünen) `Region`-Objekts, mit lokalisierten Parametern. Im gleichnamigen, globalen *Region*-Objekt werden Extruder und Flussberechnungen verwaltet.

Die Informationen über das zu druckende Objekt werden zunächst in einer mit Topologieinformationen angereicherten, internen STL-Repräsentation gespeichert. Der eigentliche Slicing-Schritt erzeugt eine Menge an Polygonen, indem jedes Facet mit allen berührenden Layern geschnitten (vergleiche Abb. 2.2b) und die resultierenden Linien im zweiten Schritt sortiert werden. Diese werden in einem Objekt pro Layer gespeichert. Besteht das Objekt aus meh-

[1] Mesh-Daten von CocoNut unter CC-BY-SA-Lizenz.

reren Materialien, enthält das Layer-Objekt mehrere Region-Unterobjekte. Informationen über Schichtdicke, zu verwendenden Extruder und Filament werden jeweils in den lokalen `Region`-Objekten gespeichert, da eine manuelle Veränderung der Schichtdicke bereits implementiert ist. Dies ist eine hilfreiche Vorraussetzung für eine spätere dynamische Zuordnung, wobei Berechnungen zum Filamentfluss weiterhin von den bereits vorhandenen Objekten durchgeführt werden können.
Für Polygonberechnungen wird die unter der Boost Software Licence stehende **Clipper** Bibliothek [8] von Angus Johnson verwendet.

3.2.2. CuraEngine

CuraEngine [6] ist ein seit Anfang 2013 von der Ultimaker B.V. entwickeltes Slicing-Werkzeug. Es ist in C++ geschrieben, enthält keine GUI und ist primär als Backend für Ultimakers 3D-Druckersoftware **Cura** gedacht. Es steht ebenfalls unter der AGPLv3-Lizenz und verwendet die **Clipper** Bibliothek [8] für Polygonberechnungen, sonst aber keinerlei externe Bibliotheken. Die durchgängige Implementierung in C++ und der (auch durch das geringe Alter) übersichtliche Code machen **CuraEngine** sehr schnell, in der Regel um Größenordnungen schneller als andere Slicer.
Variable Schichtdicken sind nicht vorgesehen, Informationen über Extruder und Filament werden zentral verarbeitet und gelten immer für das gesamte Objekt. Extruder mit unterschiedlichen Eigenschaften wie Düsendurchmesser oder Filamentdicke, sind nicht modelliert.
STL-Daten werden ebenfalls mit zusätzlichen Topologieinformationen intern gespeichert und durch einen Slicing-Schritt in Polygone überführt. Anschließend wird jedes äußere Polygon, das kein anderes Polygon berührt, als „LayerPart“ behandelt. Die weitere Pfadgenerierung erfolgt pro LayerPart. Dieses Konzept ist interessant, um etwa lokal adaptives Slicing, wie es Tyberg vorgeschlagen hat (Abschnitt 2.3.1), zu implementieren. Die LayerParts werden aber erst nach dem Slicing generiert, daher ist es nicht möglich direkt variable Schichtdicken umzusetzen.

3.2.3. Sonstige

Skeinforge ist ein prinzipiell mächtiges, in Python geschriebenes Programm [3]. Es bietet eine hohe Anzahl an Optionen und Visualierungswerkzeuge für den erzeugten *G-Code*. Durch die gewachsene Struktur ist es aber wenig benutzerfreundlich, langsam und wird seit März 2012 nicht mehr weiterentwickelt.

RepRapPro Slicer ist eine Weiterentwicklung der original RepRap Host Software durch RepRapPro Ltd. Die Software ist in Java geschrieben, verhältnismäßig langsam und wurde seit Juni 2013 nicht weiter entwickelt.

E3D ist in C geschrieben, wird seit 2011 ebenfalls nicht weiter entwickelt und hat nie einen zuverlässig funktionalen Status erreicht.

Nicht Quelloffen Es existiert eine Reihe (kommerzieller) Programme mit prinzipiell interessanten Eigenschaften. Diese sind z.T. kostenlos (Makerware [9], KISSlicer [4]) oder ausschließlich kostenpflichtig (Netfabb [7]). Da in keinem Fall der Quellcode zur Verfügung steht sind sie für diese Betrachtung nicht relevant.

3.2.4. Auswahl der Software

Für diese Arbeit wird **Slic3r** als Grundlage verwendet. Hauptgründe sind die generische Modellierung und lokale Anpassbarkeit der Druckerparameter und die bereits verankerte Unterstützung variabler Schichtdicken. Diese Aspekte überwiegen die in Teilen komplexe, gewachsene Struktur und mäßige technische Dokumentation. **CuraEngine** käme als einzige ernsthafte Alternative in Frage, ist deutlich übersichtlicher gestaltet, erfordert aber deutlich mehr Implementierung grundlegender Funktionen. Zu Beginn dieser Arbeit befand sich **CuraEngine** zudem in einem noch sehr jungen Entwicklungsstadium.

4. Implementation der Algorithmen

Einige der in Abschnitt 2.3 vorgestellten Verbesserungsansätze sind bereits implementiert bzw. auf andere Weise realisiert worden. Tyberg [Tyb98] motiviert lokal adaptives Slicing unter anderem damit, dass bei Druckvorgängen mit mehreren gleichzeitig erzeugten Objekten, das Objekt mit der höchsten Anforderung den Gesamtprozess verlangsamt. Dieses Problem ist bereits weitgehend gelöst, da sowohl in **Slic3r**, wie auch in anderen Programmen jedes Objekt einzeln verarbeitet wird. Die nachträgliche Zuordnung einzelner Polygone über ihre Lage zu einem Objekt hätte daher keinen Mehrwert und würde nur zusätzliche Rechenleistung kosten. Das Aufteilen einzelner Zweige eines Objekts (*Branching*), um jedem Zweig eine eigene Auflösung zuweisen zu können, ist weiterhin wünschenswert, wird aber in dieser Arbeit nicht weiter verfolgt, da die Daten dazu in **Slic3r** nicht nur vertikal, sondern auch horizontal partitioniert werden müssten, was eine tiefgehende Neustrukturierung der Software erforderlich machen würde.
Slic3r verwendet im Normalfall eine globale Schichtdicke, auch wenn mehrere Objekte verarbeitet werden. Es ist zwar möglich, jedem Objekt manuell seine Schichtdicke zuzuweisen, dies ist allerdings umständlich und löst das Problem daher nicht sauber. Intern wird aber jedes Objekt einzeln behandelt und eine individuelle Menge an Schichten generiert. Die im Folgenden beschriebene, adaptive Berechnung der Schichtdicken, ist daher verhältnismäßig einfach pro Objekt implementierbar und macht eine zusätzliche Interaktion des Benutzers überflüssig.

Die von Sabourin [Sab96] vorgeschlagene Trennung in „Exteriour“ und „Interiour“ ist bereits weitgehend implementiert, allerdings mit einem etwas anderen Lösungsansatz. Es wird nicht mit *Slabs* gearbeitet, die später verfeinert werden, sondern der umgekehrte Ansatz einer späteren Vergröberung verwendet. Zunächst wird das gesamte Objekt mit kleiner Schichtdicke gesliced. Die so generierten Polygone werden im zweiten Schritt durch Offsetting in Außenbereich (*Perimeter*) und Innenbereich (*Infill*) unterteilt. Im dritten Schritt werden jeweils n Infillebenen zu einer dicken Ebene zusammengefasst. Dieses Konzept wird als *Microlayering* bezeichnet. Die Anzahl der jeweils zusammenzufassenden Ebenen n kann frei gewählt werden, ist aber durch den Durchmesser des eingesetzten Extruders begrenzt.

Multimaterialobjekte, insbesondere für mehrfarbigen Druck, werden in **Slic3r** auf Schichtebene wie Einzelobjekte behandelt. Es werden zunächst die Schichten festgelegt und in jeder Schicht

für jedes Unterobjekt ein `Region`-Objekt verwendet. Jedem Unterobjekt wird ein Extruder zugeordnet, das Verwenden zweier Extruder mit unterschiedlichem Durchmesser für die horizontale Auflösung in einem Objekt ist daher nicht möglich. Für mehrfarbigen Druck werden i.d.R. Extruder mit gleichem Durchmesser eingesetzt.

4.1. Adaptives Slicing

4.1.1. Schichthöhe nach Objektoberfläche

Als Grundlage für die Berechnung der Schichtdicke wurde zunächst der *Cusp*-Algorithmus nach Dolenc und Mäkelä [DM94] implementiert, wie er in Abschnitt 2.3.1 beschrieben ist. Dazu wurde eine eigene Klasse `AdaptiveSlicing` erstellt, in der die Funktionalität gekapselt ist. Der Prozess ist zustandsbehaftet, im jeweils nächsten Schritt kann nur die Schichtdicke einer Ebene errechnet werden, die mindestens auf der gleichen Höhe liegt wie die letzte. Da die Modelldaten als Menge von Dreiecken vorliegen, kann die *cusp height* für jedes Facet berechnet werden, die Auswahl von Punkten entfällt hier.
Um die Berechnung effizient zu gestalten, wird zunächst eine Liste der Minimum- und Maximumwerte aller Facets des Modells erzeugt. Diese wird aufsteigend nach den Minimumwerten sortiert. Falls nicht vorhanden, werden zusätzlich die Oberflächennormalen für jedes Facet erzeugt. Es wird eine Referenz `current-facet` auf das letzte Dreieck gehalten, dass sich vollständig unterhalb der betrachteten Ebene befindet. Somit muss der bereits analysierte Teil des Objekts nicht mehr durchlaufen werden. Gegeben eine Schnittebene `slice-z`, lassen sich die für einen Schnitt relevanten Facets finden durch:

```
1:  FindFacets(slice-z)
2:  facet ← current-facet
3:  result[]
4:  while facet.max-z < slice-z do
5:      facet++
6:  end while
7:  current-facet ← facet
8:  while facet.min-z < slice-z do
9:      if facet.max-z > slice-z then
10:         result.append(facet)
11:         facet++
12:     end if
13: end while
14: return result
```

Im optimalen Fall werden damit tatsächlich nur die Facets betrachtet, die die Schnittebene berühren. Im ungünstigsten Fall (ein Facet erstreckt sich über die gesamte Höhe des Objekts), ist das Vorsortieren wirkungslos und es werden alle Facets betrachtet.

Slic3r unterscheidet für den Slicing-Prozess zwischen zwei Höhen: `print-z` und `slice-z`. `Print-z` bezeichnet dabei die Obergrenze des aktuellen Layers, auf dieser Höhe bewegt sich der Extruder. `Slice-z` ist die Ebene, in der der Schnitt mit dem Modell berechnet wird. In **Slic3r** ist dies auf halber Höhe des Layers (Grüne Ebene in Abb. 4.1). Somit ergibt sich unabhängig von der Orientierung der Oberflächenneigung ein gleichmäßiges Fehlvolumen um die Solloberfläche herum.

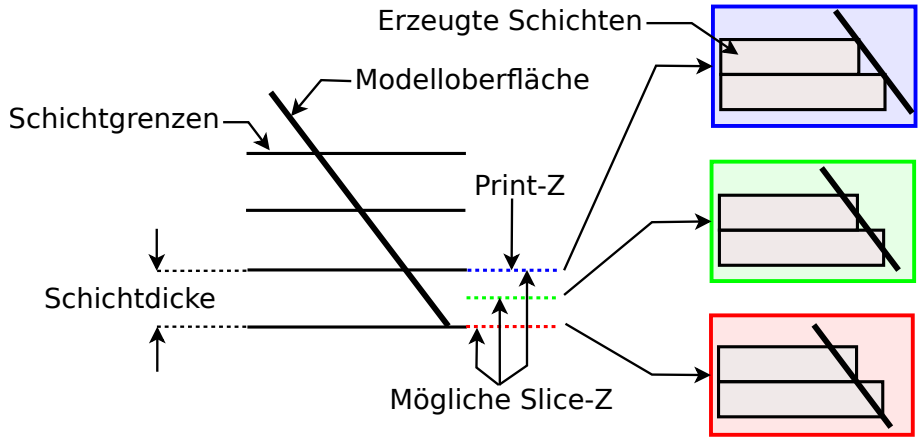
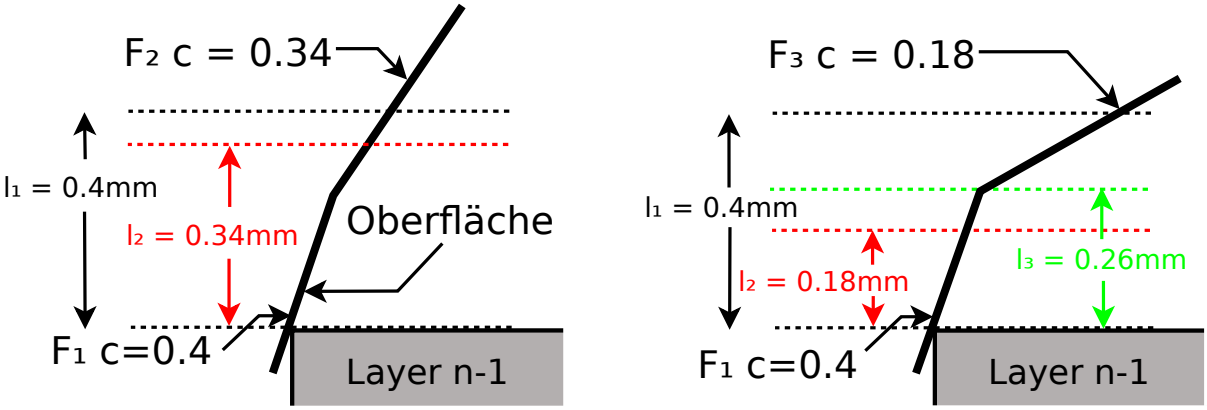


Abbildung 4.1. – Mögliche Schnittebenen im Slicing-Prozess. Während des Drucks befindet sich der Extruder immer auf der Höhe Print-Z. Je nachdem, auf welcher Höhe die Slicing-Software den Schnitt mit dem Objekt berechnet, ergibt sich bei geneigten Oberflächen ein außen oder innen liegendes Fehlvolumen.

Die *cusp height* c, und somit die vorläufige Schichtdicke, wird aus den Normalen alle gefundenen Facets mit Gleichung (2.2) berechnet. Die Verwendung der tatsächlichen Schnittebene `slice-z` ist nur dann sinnvoll, wenn Änderungen der Oberfläche ober- und unterhalb der Schnittebene ignoriert werden sollen. Daher wird c zunächst auf Basis der Untergrenze des zu berechnenden Layers (rote Ebene in Abb. 4.1) berechnet. Dies ist genau die Druckebene `print-z` des vorigen Layers. Träte innerhalb der in diesem Schritt errechneten Höhe ein Facet (F_2) mit stärkerer Neigung auf (Beispiel in Abb. 4.2a), würde die vorgegebene maximale Abweichung C_{max} überschritten werden. Um dies zu vermeiden, werden im zweiten Schritt alle Facets innerhalb des im ersten Schritt ermittelten Layers betrachtet, und c gegebenenfalls auf den neuen Wert reduziert.
Dies kann dazu führen, dass ein stark geneigtes Facet (F_3) am oberen Rand des Layers die gesamte Schichtdicke unnötig verkleinert (Beispiel in Abb. 4.2b). Um dies zu verhindern, wird anstelle des neuen c der Abstand zwischen Untergrenze des Layers und Unterkante des Facets $l = F_3\texttt{.min-z} - \texttt{slice-z}$ verwendet.



(a) Die *cusp height* von F_2: $c = 0.34$ liegt höher als der unterste Punkt des Facets: F_2.min-z $= 0.26$. c wird zu l_2 verringert.

(b) Die *cusp height* von F_3: $c = 0.18$ liegt niedriger als der unterste Punkt des Facets: F_3.min-z $= 0.26$. c wird zu l_3 verringert.

Abbildung 4.2. – Anpassung der im ersten Schritt ermittelten *cusp height* (l_1), aufgrund sich in diesem Layer befindlicher Facets mit stärkerer Neigung.

Die Dicke des nächsten Layers wird mit diesen Erweiterungen berechnet durch:

1: $c \leftarrow min$(Cusp(FindFacets(z)))
2: **while** facet.min-z $<$ $z + c$ **do**
3: $c_1 \leftarrow$Cusp(facet)
4: **if** $c_1 < c$ **then**
5: $c \leftarrow max(c_1,$ (facet.min-z$-z$))
6: **end if**
7: facet++
8: **end while**
9: **return** $min(max(c, L_{min}), L_{max})$

Wobei z die Unterkante des zu berechnenden Layers, L_{min} und L_{max} die minimal und maximal mögliche Schichtdicke des Extruders sind.

Horizontale Facets erzeugen eine *cusp height* von $c = C_{max}$. Da horizontale Flächen häufig kritische Elemente eines Objekts sind, wird für solche Facets immer die Z-Distanz verwendet. Mit diesen Erweiterungen berechnet der *Cusp*-Algorithmus immer eine maximale Schichtdicke, die eine Abweichung kleiner als C_{max} erzeugt.

Die Berechnung der Schichtdicke für Multimaterialobjekte erfolgt mit dem gleichen Algorithmus. Sie bestehen aus mehreren getrennten *Mesh*-Objekten, ein Layer zieht sich aber immer durch alle Unterobjekte. Jedem *Mesh*-Objekt ist ein Extruder zugeordnet, diese können unterschiedliche Durchmesser haben. Für die Bestimmung der Schichtdicke in solchen Objekten

werden L_{min} und L_{max} deshalb auf den kleinsten bzw. größten für alle Extruder zulässigen Wert gesetzt. Für jedes *Mesh*-Objekt wird ein `AdaptiveSlicing`-Objekt erzeugt und für jeden Layer das Minimum aller errechneten Schichthöhen verwendet.

Wie in Abschnitt 2.2 angedeutet, gibt es grundsätzlich 2 Wege die Schnittoperationen durchzuführen:

- Schnitt eines Layers mit allen Facets berechnen (Abb. 2.2a).
- Schnitt eines Facets mit allen Layern berechnen (Abb. 2.2b).

Beide Varianten lassen sich parallelisieren, entweder indem jedem Prozess eine Menge von Layern, oder jedem Prozess eine Menge von Facets zugeordnen wird. In **Slic3r** ist die 2. Variante implementiert. Beides funktioniert nur, wenn die Positionen aller Layer vor der Berechnung bekannt sind. Bei statischem Slicing ist dies trivialerweise der Fall. Bei adaptivem Slicing ist es prinzipiell ebenfalls möglich, im ersten Schritt die Oberfläche vollständig zu analysieren und die Positionen aller Layer festzulegen. Die Wahl eines Extruders kann, wie in Abschnitt 4.2.2 beschrieben, aber möglicherweise erst durchgeführt werden, wenn die Polygone eines Layers vorliegen. Die Höhe eines Layers wird aber durch den Durchmesser des Extruders beschränkt. Diese wechselseitige Abhängigkeit schließt Parallelisierung an dieser Stelle weitgehend aus. Praktisch wird die bereits existierende `slice`-Methode in Slic3r jeweils nur mit dem aktuellen Layer aufgerufen, anstatt mit einer Liste von Layern. Dies serialisiert den Ablauf.

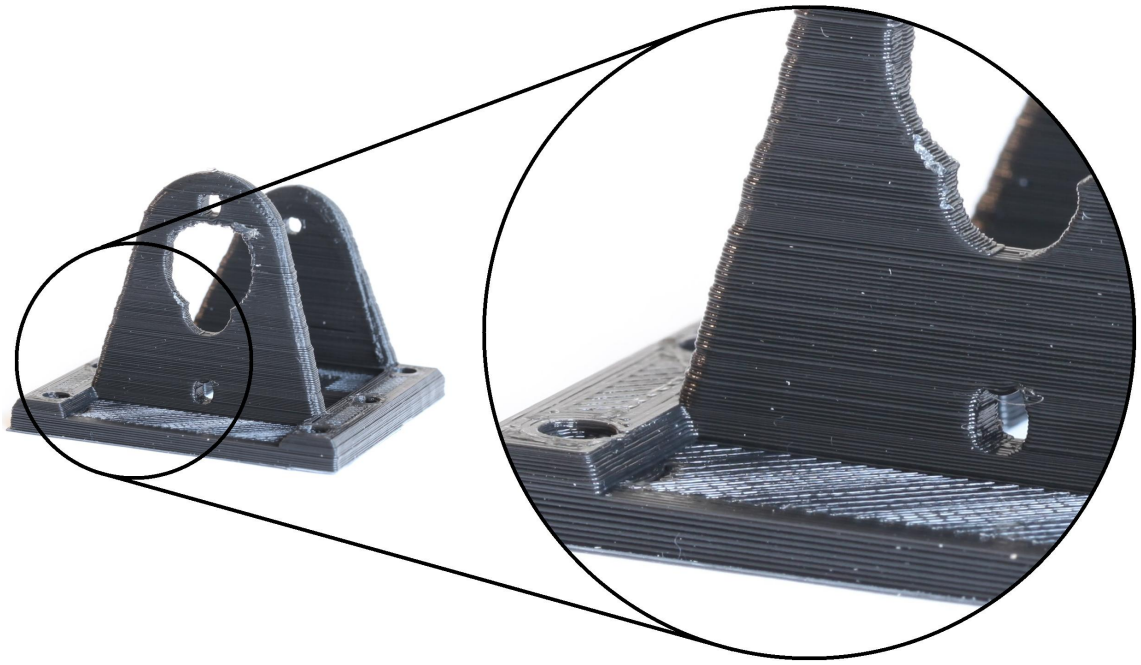


Abbildung 4.3. – Beispiel eines mit adaptiver Schichtdicke gedruckten Objekts. In der Vergrößerung ist deutlich die Verringerung der Schichten an schrägen Oberflächen und in den Rundungen zu erkennen.

Für Endnutzer, die sich nicht intensiver mit der Thematik beschäftigt haben, ist nicht unmittelbar offensichtlich, wie sich eine Änderung des *Cusp*-Wertes C_{max} auf das Druckergebnis auswirkt. Soll der Wert vom Benutzer direkt beeinflussbar sein, erleichtert es die Benutzung daher deutlich, wenn sinnvolle Einschränkungen gemacht werden, oder die Auswirkung sogar grafisch dargestellt wird.

- Ein Wert $C_{max} > L_{max}$ ist nicht sinnvoll, da bereits bei $C_{max} = L_{max}$ unabhängig von der Objektoberfläche mit maximaler Schichtdicke gedruckt wird.

- Für sehr kleine Werte $C_{max} \in [0, L_{min}]$ verläuft die Höhenverteilung nicht mehr glatt, sondern „knickt“ bei L_{min} ab, Beispiel in Abb. 4.4, rote Linie.

- Für $C_{max} > L_{min}$ ist die geringste verwendete Schichtdicke C_{max} und damit größer als L_{min}. Die technisch kleinste mögliche Auflösung wird nie ausgenutzt.

Der Verlauf der Schichtdicke im Intervall $[0, L_{max}]$ ist wie in Abb. 4.4 dargestellt nicht linear. Eine Begrenzung auf dieses Intervall wurde in **Slic3r** implementiert. Ob eine mit den aktuell eingestellten Parametern erzeugte Kurve im Stil von Abb. 4.4 in der Benutzeroberfläche für Benutzer ohne Vorwissen hilfreich wäre, wurde bislang nicht untersucht.

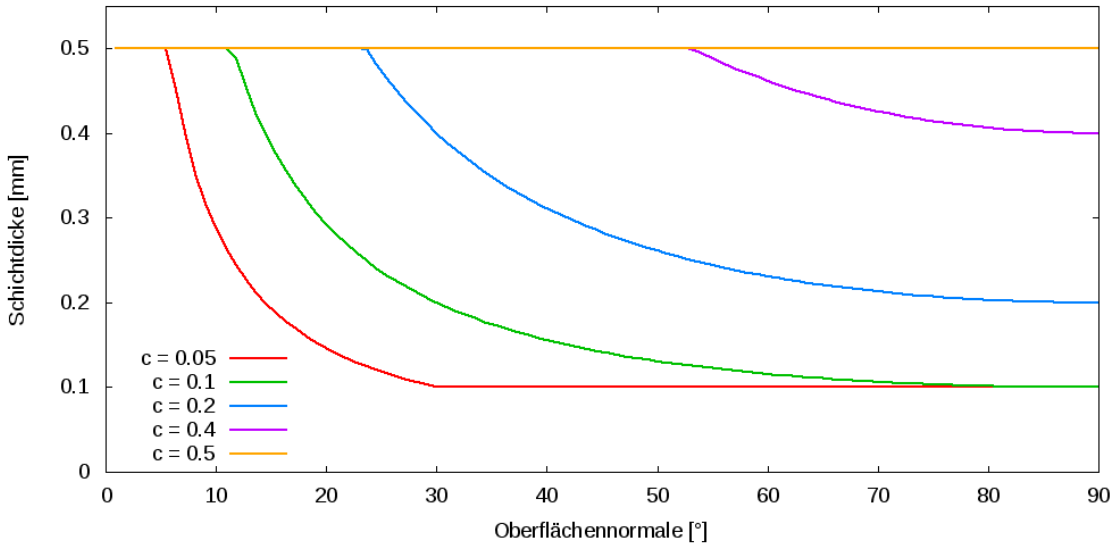


Abbildung 4.4. – Dicke der erzeugten Schichten in Abhängigkeit von der Oberflächennormale. Die Schichtdicke ist in dieser Abbildung durch die Extrudereigenschaften begrenzt auf das Intervall [0.1, 0.5].

4.1.2. Horizontale Oberflächen

Wie in Abschnitt 2.3.1 und Abbildung 2.3 beschrieben, ergeben sich durch den Slicing-Prozess im Allgemeinen Fehler an horizontalen Oberflächen eines Objekts. Der *Cusp*-Algorithmus verringert zwar die Abweichung, aber nur bis zur minimal möglichen Schichtdicke L_{min}. Die Abweichung wird dadurch im Mittel kleiner, tritt aber weiterhin auf. Im Zuge der oben beschriebenen Erweiterungen des Verfahrens zur Berechnung der Schichtdicke, lässt sich dieses Problem mit wenig Aufwand ebenfalls lösen. Das `AdaptiveSlicing`-Objekt wurde dazu um eine Funktion erweitert, die den Abstand zur nächsten horizontalen Fläche ermittelt. Aus der sortierten Liste aller Facets wird dazu eine zweite, ebenfalls sortierte Liste aller horizontalen Facets abgeleitet. Ein Pointer hält die Position des letzten Eintrags unterhalb des aktuell betrachteten Intervalls. Beim Aufruf wird eine maximale Vorwärtsdistanz d_z angegeben, der Suchraum beschränkt sich somit auf die Facets, die sich tatsächlich innerhalb des gesuchten Bereichs befinden. Der Suchraum ist nach oben begrenzt durch die Höhe des Objekts.

Um Layer zu erzeugen, deren Oberfläche sich genau auf Höhe einer horizontalen Fläche des Objekts befindet, wird in jeder Iteration der Bereich direkt oberhalb des aktuellen Layers betrachtet. Die Überprüfung auf horizontale Oberflächen findet nach der Ermittlung der Schichtdicke durch den Cusp-Algorithmus statt. Als Suchintervall genügt [`print-z`, `print-z` $+ L_{min}$], also genau eine technisch minimale Schichtdicke oberhalb des aktuellen Layers. Tritt in diesem Bereich keine horizontale Fläche auf, ist in der nächsten Iteration mindestens eine weitere Schicht der Dicke L_{min} möglich. Der aktuelle Layer muss nicht betrachtet werden, da eine Fläche unterhalb von L_{min} bereits in der vorigen Iteration erkannt worden wäre und eine Fläche oberhalb von L_{min} bereits durch den *Cusp*-Algorithmus korrekt getroffen wird.

Wurde im Suchintervall eine horizontale Fläche gefunden, wird versucht, die Dicke des aktuellen Layers zu verringern, um die Qualität nicht herabzusetzen. Ist dies nicht möglich (weil die Schichtdicke dann unter L_{min} fallen würde), wird der Layer bis zur horizontalen Fläche vergrößert. Zwei horizontale Flächen mit einer Distanz kleiner L_{min} können nicht korrekt wiedergegeben werden.

4.2. Extrusionsbreite

In Abschnitt 2.3.2 wurde ein Ansatz unter Verwendung einer zweistufigen Düse vorgestellt. Dieser Ansatz ist technisch verhältnismäßig komplex. Einige aktuelle low-cost Drucker verfügen aber über zwei oder mehr Düsen. Werden diese mit dem gleichen Material bestückt, haben aber unterschiedliche Durchmesser, kann je nach lokaler Anforderung der Passende eingesetzt werden. Zusätzlich ist es möglich, durch Steigern oder Verringern der Flussgeschwindigkeit die Extrusionsbreite in gewissen Grenzen zu variieren. Im Folgenden wird ausgeführt, wie der geeignete Extruder gewählt werden kann. Dabei findet ein Extruderwechsel zunächst nur beim Übergang von einem Layer zum Nächsten statt.

Der Umriss einer Schicht eines zu druckenden Objekts, repräsentiert durch eine Menge von Polygonen, kann durch einen kreisförmigen Extruder im Allgemeinen nur approximiert werden. Ob eine Schicht mit einem bestimmten Extruder korrekt wiedergegeben kann, lässt sich mit Hilfe des Konzepts der r-regularität überprüfen. R-regularität wurde eingeführt von Serra [Ser82]. Sei B der Einheitskreis. Eine Menge X heißt *r-regulär*, wenn sie morphologisch offen und geschlossen ist, bezogen auf eine Scheibe mit Radius $r > 0$:

$$X = (X \ominus rB) \oplus rB = (X \oplus rB) \ominus rB \tag{4.1}$$

Für den vorliegenden Anwendungsfall genügt eine schwächere Definition, die sich nur auf den Innenbereich des Umrisses beschränkt:

$$X = (X \ominus rB) \oplus rB \tag{4.2}$$

Lemma 1. *Der Umriss eines zu druckenden Objekts kann von einem kreisförmigen Extruder mit Radius r_e grundsätzlich nur dann korrekt abgebildet werden, wenn er* r-regulär *ist für $r \geq r_e$.*

Daraus folgen zwei wesentliche Eigenschaften [BA92]. Der Krümmungsradius ist an jedem Punkt des Umrisses größer oder gleich r. Der Radius eines größten leeren Kreises (Largest Empty Circle (LEC)) ist niemals kleiner als r. Das bedeutet erstens, dass ein Extruder mit Radius $r_e \leq r$ den Umriss an jeder Stelle abfahren kann und es zweitens keine Stelle gibt, die er nicht mindestens einmal vollständig passieren kann. Der erste Punkt, Kurvenradien kleiner r_e, wurde am Beispiel spitzer Winkel von [BRA$^+$] behandelt, siehe Abschnitt 2.3.2.
Im Folgenden wird davon ausgegangen dass $r = r_e$, der Extruderradius also dem Definitionsradius der r-Regularität entspricht. Vielfache davon werden als xr-regulär bezeichnet, beispielsweise 2r-regulär mit $r = 2 \cdot r_e$.

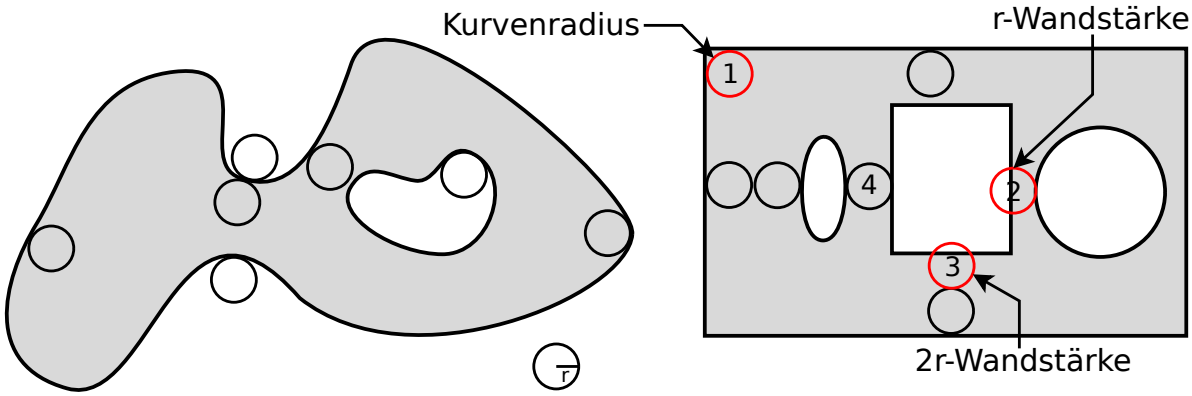


Abbildung 4.5. – **Links**: r-regulärer Umriss. **Rechts**: nicht r-regulärer Umriss. Der Kurvenradius der Winkel ist zu klein (1. Fall). Die Wandstärke auf der rechten Seite des Objekts ist zu gering (2. Fall). Die Wandstärke im unteren Bereich erfüllt r-Regularität, aber nicht für 3D-Drucker relevante 2r-Regularität.

Auch r-reguläre Umrisse können nur in Sonderfällen korrekt wiedergegeben werden. Erstens, weil ein „Winkel“, auch wenn er rund genug ist, vom Extruder betreten und wieder verlassen werden muss. Zweitens weil ein „Steg“ im Objekt (im folgenden Wand) zwar durch genau einen Extruderdurchlauf abgebildet werden kann (Abb. 4.5, #4), eine Wand mit Durchmesser $r_e < r < 2r_e$ aber schmaler oder breiter wird, da sie entweder durch einen oder 2 Durchläufe erzeugt werden muss. Lemma 1 kann daher spezifischer ausgedrückt werden als:

Lemma 2. *Der Umriss eines zu druckenden Objekts kann von einem kreisförmigen Extruder mit Radius r_e immer dann korrekt abgebildet wenn er mindestens 2r-regulär ist.*

Damit kann in jedem Fall der Umriss korrekt erzeugt werden, für das Füllen des Objekts reicht aber auch diese Bedingung nicht aus. Im Fall $2r_e < r < 3r_e$ ist zwischen den Konturpfaden (*Perimeter*) nicht genug Raum für einen weiteren Fülldurchlauf. Wände mit $r \geq 3r_e$ können prinzipiell vollständig gefüllt werden, wenn das Füllmuster entsprechend erzeugt wird.

Um einen Extrusionspfad zu erzeugen, wird jeder Layer zunächst in einen Außenbereich (*Perimeter*) und einen Innenbereich (*Infill*) unterteilt. Der Perimeterbereich wird erzeugt, indem die Kontur des Polygons abgefahren wird. Mehrere Zyklen werden durch wiederholtes Offsetting des Polygons erzeugt, die Anzahl der Zyklen p ist durch den Benutzer einstellbar. Um festzustellen, ob der Perimeter eines Polygons P durch einen Extruder mit Radius r_e mit p Zyklen dargestellt werden kann, wird nach Definition der r-Regularität nacheinander $X \ominus r_e pB$ und $X' \oplus r_e pB$ durchgeführt. Dies entspricht einem zweifachen Offseting des Polygons P um $2r_e p$ mit wechselndem Vorzeichen zu P''. Sind P und P'' äquivalent, ist das Polygon darstellbar. Diese Operation entspricht dem Offset-Typ `jtRound` in Abb. 4.7a.

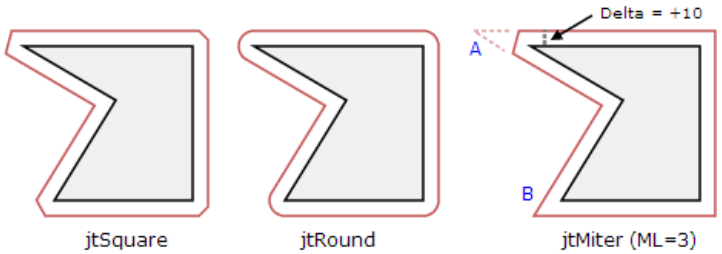
Diese Bedingung ist zu strikt, da für Winkel eine gewisse Toleranz gelten soll. Ein besserer Ansatz ist daher *mitered offsetting*. Dieser Ansatz wurde von Park und Chung [PC03] für abtragende NC-Verfahren eingeführt. Die Grundidee dabei ist, dass spitze Ecken auch nach dem Offsetting spitz bleiben. Da die resultierende Spitze bei kleinen Winkeln sehr lang werden kann, wird sie bei Überschreiten eines gewissen Grenzwerts abgeschnitten. Park und Chung schneiden jeden Winkel bei bei einer Winkelöffnung $> 1.5\pi$. In der **Clipper** Bibliothek ist der Grenzwert definiert als der Abstand zwischen der Spitze des Polygonwinkels und der Spitze des Offsetwinkels, und wird als Faktor $miterLimit$ über dem Offsetwert Δ angegeben, um den das gesamte Polygon versetzt werden soll. Ist die Distanz des Winkels größer als $miterlimit \times \Delta$, wird der Winkel geschnitten.
Park und Chung berechnen den *mitered offset* direkt mit dem von ihnen erweiterten, etwas komplexen, Pairwise Interference Detection (PWID)-Ansatz. Eine effizientere Strategie ist die Verwendung von *straight skeletons*, für deren Berechnung mittlerweile Algorithmen mit vertretbarer Laufzeit existieren [Hub12].

Abbildung 4.6. – Verlauf des *mitered offset* bei einem internen Winkel $\leq 1.5\pi$ (links) und $\geq 1.5\pi$ (rechts). Abbildung aus [PC03, Fig. 4.]

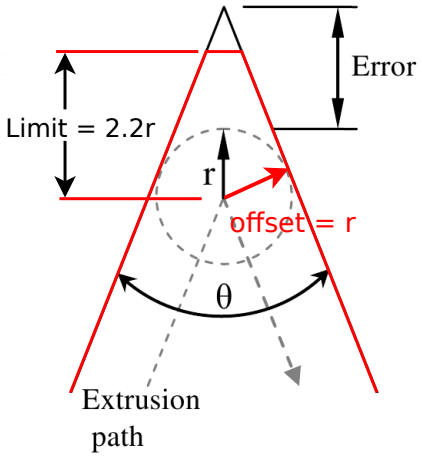
Für die Bestimmung der Fehlergrenze, oberhalb derer, wenn möglich, eine kleinere Extrusionsbreite verwendet werden soll, wird das von [BRA$^+$] eingeführte Fehlermaß E aus Gleichung (2.5) verwendet. Das Polygon soll somit genau dann geschnitten werden, wenn die Distanz $\Delta \times miterLimit$ größer ist als $E + r$ (Abb. 4.7b). Mit $\Delta = r_e$ ergibt sich für die Offsetoperation:

$$miterLimit = \frac{E + r_e}{r_e} \tag{4.3}$$

Dabei ist das tatsächlich für das Offsetting verwendete Δ, wegen der Eigenschaften ähnlicher Dreiecke, unerheblich. Die Eigenschaft bleibt also auch dann erhalten, wenn $\Delta = 2r_e p$, der Extruderradius geht nur in die Berechnung von $miterLimit$ ein.
Der Polygonwinkel wird je nach Implementierung nicht zwingend bei einer Distanz von $E + r$ abgeschnitten. Um herauszufinden, ob der Winkel mit der geforderten Genauigkeit darstellbar ist, ist dies auch nicht relevant. Es reicht, festzustellen, ob überhaupt ein Schnitt stattgefunden hat und somit die 2r-Regularität verletzt wurde.



(a) Anwendung verschiedener Offset-Typen (Abb. aus Clipper Dokumentation [8]).



(b) Effekt der *mitered offset* Operation mit kleinerem $miterLimit$ als für den Fehler nötig.

Abbildung 4.7.

In der Implementierung wurde $E = 0.4$ als Voreinstellung verwendet. Gängige Extruderdurchmesser bewegen sich im Bereich 0.25mm-0.5mm. Wie in Abb. 4.8 ersichtlich, wird bei diesem Fehler für Winkel unterhalb ∼45°ein Extruder mit kleinerem Durchmesser als 0.5mm gewählt.

Die in **Slic3r** implementierte Berechnung erfolgt in zwei Schritten. Es werden maximal zwei Extruder berücksichtigt, da die zu erwartende Verbesserung durch 3 oder mehr Extruder im Verhältnis zum Aufwand gering ist. Es reicht daher aus, die Überprüfung für den größeren Extruder e_1 durchzuführen. Ist dessen Durchmesser für das vorliegende Polygon zu groß, wird der kleinere Extruder e_2 verwendet.
Aus P wird durch Offsetting mit $\Delta = 2r_{e1}p$ das Zwischenergebnis P' errechnet. Ist die Anzahl der resultierenden Polygone (Außenkontur + Innenkonturen) verändert, gibt es eine Wand die nicht abgefahren werden kann (LEC $< r$). Die Berechnung kann bereits jetzt abgebrochen und e_2 verwendet werden. Andernfalls wird der zweite Offset-Schritt mit $-\Delta$ durchgeführt und die Differenz aus P und P'' gebildet. Ist die Fläche des Ergebnisses größer als ein Toleranzschwellwert ε, wird e_2 verwendet, sonst e_1. Der Toleranzschwellwert ist notwendig, da die Clipping-Operation bei manchen Polygonen sehr kleine Abweichungen erzeugen kann.

Bei statischem Slicing darf die verwendete Schichtdicke im gesamten Objekt nicht größer sein als der Durchmesser des kleineren Extruders. Bei adaptivem Slicing wird zunächst die Schichtdicke für den ersten Extruder berechnet und der Polygonsatz erzeugt. Mit diesen Daten wird die Extrusionsbreite geprüft und bei einem Extruderwechsel die Berechnung der Schichtdicke mit neuen Grenzwerten wiederholt.

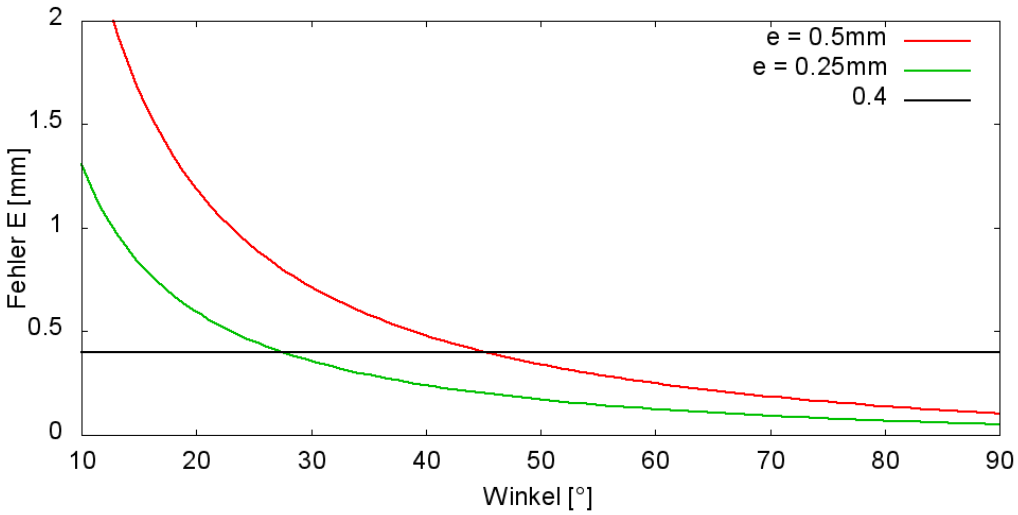


Abbildung 4.8. – Fehler E, der durch die Approximation spitzer Winkel mit runden Extrudern entsteht, dargestellt für Extruderdurchmesser 0.5mm und 0.25mm. Die voreingestellte Grenze, unterhalb derer in **Slic3r**, wenn möglich, ein kleinerer Extruder verwendet wird, ist schwarz eingezeichnet.

4.2.1. Variable Extrusionsbreite

Die Breite der aufgetragenen Kunststoffspur kann in gewissen Grenzen variiert werden, indem das Volumen des pro Strecke vorgeschobenen Kunststoffs verändert wird. Das Extrudat „quillt“ dann entweder um die Düse herum und wird breiter, aber weniger glatt, oder es zieht sich im Zentrum der Düse zu einem schmaleren Bereich zusammen. Dabei kommt es mit abnehmendem Volumen zunehmend häufiger vor, dass der Faden reißt oder sich die Stabilität des Objekts verringert. Der maximal mögliche Wert des Extrusionsfaktors f_e hängt mindestens vom Extruder, dem Düsendurchmesser und der Zusammensetzung des Filaments ab. Es wurde eine kurze Testreihe durchgeführt, um eine Größenordnung für anwendbare f_e zu bestimmen (Siehe Anhang A.2). Bis zu einem Wert von $f_e \approx 1.5$ wurden unter allen Bedingungen akzeptable Ergebnisse erzielt, mit einigen Extruder-/Filamenttypen waren deutlich größere Werte möglich. Im Folgenden ist f_e daher konstant 1.5 und wurde auch in **Slic3r** als Konstante implementiert.

Diese Eigenschaft kann genutzt werden, um Extruderwechsel zu vermeiden, wenn der Durchmesser von E_1 nur geringfügig zu groß ist und um qualitativ bessere Perimeter zu erzeugen, die Zwischenräume vollständig ausfüllen.
Die optimale Perimeterbreite zu berechnen ist ein Optimierungsproblem, bei dem der maximale, 2r-Regularität erhaltende, Offset gesucht ist. Da keine analytische Lösung dieses Problems bekannt ist, wurde eine binäre Suche implementiert. Diese beginnt mit $\Delta = 2r_{e1}p \times f_e$ und

terminiert entweder im ersten Schritt, wenn der maximale Offset möglich ist oder bei Unterschreiten eines Fehlerschwellwertes ε:

```
1: MaxOffset(f_e, ε)
2: Δ ← 2r_e1 p f_e
3: d ← Δ/2
4: if not r-regular(Δ, E) then
5:    Δ ← Δ − d
6:    repeat
7:       d ← d/2
8:       if r-regular(Δ, E) then
9:          Δ ← Δ + d
10:      else
11:         Δ ← Δ − d
12:      end if
13:   until d < ε
14: end if
15: return Δ
```

Mit der so berechneten Perimeterbreite kann im Anschluss die optimale Extrusionsbreite (`extrusion_width`) ermittelt werden. Dabei können sich die möglichen Intervalle bei zwei Extrudern durchaus überschneiden. Das Intervall beträgt für einen Extruder:

$$I_e = [\frac{2r_e}{f_e},\ \ldots,\ f_e \times 2r_e] \tag{4.4}$$

Bei einem Durchmesser von 0.5mm also: $[0.33,\ 0.75]$, bei 0.25mm: $[0.17,\ 0.375]$.
Passt eine mögliche Extrusionsbreite weniger als p mal in die Perimeterbreite Δ, wird der kleinere Extruder verwendet. Ist dies nicht mehr möglich, wird p soweit reduziert wie nötig:

```
1: if Δ/p ∈ I_e1 then
2:    return (E_1, width ← Δ/p)
3: else
4:    if Δ/p ∉ I_e2 then
5:       p ← ⌊2r_e2 / f_e⌋
6:    end if
7:    return (E_2, width ← Δ/p)
8: end if
```

Slic3r unterstützt bereits das Konzept mehrerer Extruder. Für jeden Extruder wird eine Instanz des `Extruder`-Objekts erzeugt, in dieser sind Parameter, wie bereits extrudiertes Filament oder der Zustand der Retraction, gespeichert. Zusätzlich werden `extrusion_width` und `spacing` in einem separaten `Flow`-Objekt modelliert, da sich diese Parameter für verschie-

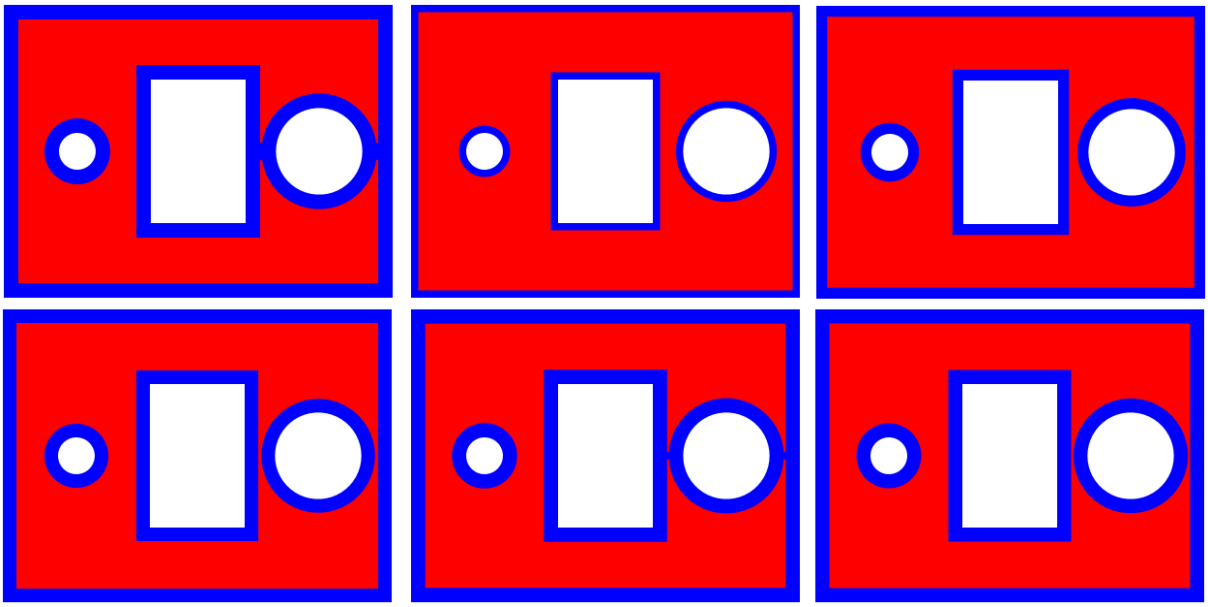


Abbildung 4.9. – Durchlauf der binären Suche zum finden der maximal möglichen Perimeterbreite.

dene Bereiche eines Objekts ändern. `Flow`-Objekte existierten bislang aber nicht pro Layer[1], sondern nur jeweils eines für den gesamten *Perimeter*-, *Infill*-, *Support*-, und *Raft*-Bereich. Da sich `extrusion_width` und `spacing` durch die adaptive Extrusionsbreite in jedem Layer verändern, wurde das `Flow`-Objekt für jeden Layer geklont und lokal vorgehalten. Die Zuordnung der `Extruder`-Objekte geschieht pro Layer und Bereich durch das Setzen einer Referenz auf das globale Objekt, das für diesen Bereich verwendet werden soll.

Bei der Umsetzung der Implementierung ergab sich eine zusätzliche Schwierigkeit aus der Art, wie die Extrusionsbreite und das Extrusionsvolumen in Slic3r berechnet werden. Da alle in dieser Arbeit vorgeschlagenen Erweiterungen optional verfügbar sein sollen, musste die Berechnung an das existierende Konzept angepasst werden. Der extrudierte Kunststofffaden ist kein Rechteck, sondern grob ein Rechteck mit Rundungen an beiden Seiten. Die `extrusion_width` gibt die Distanz von einer Außenkante des Fadens zur Anderen an. Werden zwei Fäden mit einem Abstand von `extrusion_width` nebeneinander gelegt, wäre das Volumen nicht vollständig ausgefüllt (Abb. 4.10, grün schraffierte Fläche). Daher wird für die finale Positionierung des Extrusionspfades der Parameter `spacing` verwendet, der auf dem extrudierten Volumen basiert. Die Fäden sollen dadurch leicht überlappen und in der Summe ein vollständig geschlossenes Volumen ergeben.

[1] Dieses Verhalten ist offensichtlich ein Fehler in der Implementierung. Da in Slic3r Schichthöhen für einzelne Bereiche des Objekts definiert werden können, müsste das `Flow`-Objekt auf Schichtebene verwendet werden. Im `Flow`-Objekt wird der Parameter `extrusion_spacing` berechnet, dieser hängt u.A. von der Schichdicke ab, und definiert den Abstand der extrudierten Kunststoffäden zueinander (s.U.). Der Fehler wurde gemeldet (Issue #1495 im Github-Repository [10]) aber bislang nicht behoben.

Bei einer gegebenen Schichtdicke h, ist `spacing` $= \frac{A}{h}$, wobei A die Querschnittsfläche des Extrudats ist (rot schraffierte Fläche). A ist die Differenz aus $A_{rechteck} = h \cdot extrusion\_width$ und A_1 (grüne Fläche). Mit

$$A_1 = h^2 - h^2 \cdot \frac{\pi}{4} = h^2 \cdot (1 - \frac{\pi}{4}) \tag{4.5}$$

ergibt sich für `spacing`:

$$spacing = \frac{A}{h} = \frac{A_{rechteck} - A_1}{h} = extrusion\_width - h \cdot (1 - \frac{\pi}{4}) \tag{4.6}$$

Da die durch das Offsetting berechnete Extrusionsbreite in **Slic3r** gerade dem Parameter `spacing` entspricht, wird die für den Volumenfluss benötigte `extrusion_width` berechnet mit:

$$extrusion\_width = spacing + h \cdot (1 - \frac{\pi}{4}) \tag{4.7}$$

Diese Berechnung gilt nur, wenn die `extrusion_width` größer ist als der Extruderdurchmesser + Schichtdicke. Ist sie kleiner, wird davon ausgegangen dass die seitlichen Rundungen weniger stark ausgeprägt sind, `extrusion_width` wird dann berechnet mit:

$$extrusion\_width = \frac{spacing - 2r_e \cdot (1 - \frac{\pi}{4})}{\frac{\pi}{4}} \tag{4.8}$$

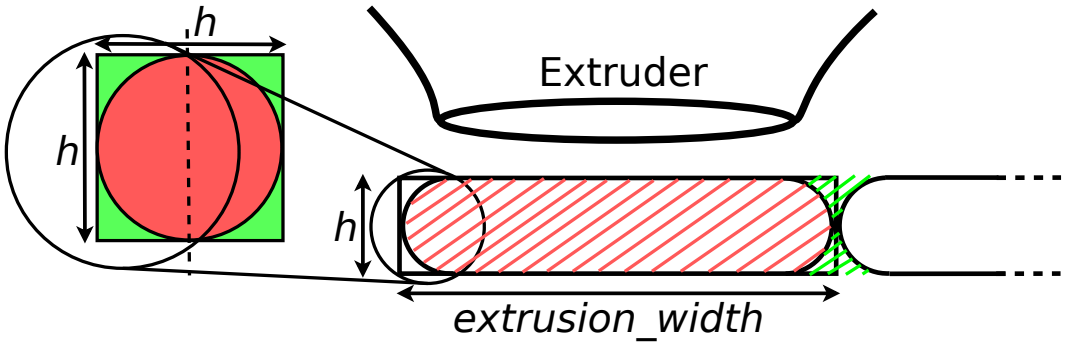


Abbildung 4.10. – Modell der Geometrie des extrudierten Kunststofffadens (rot schraffiert). Um das nicht ausgefüllte Volumen zwischen den Fäden (grün schraffiert) zu vermeiden, wird es herausgerechnet.

4.2.2. Infill

Im Innenbereich ist es grundsätzlich sinnvoll, einen möglichst großen Extruder zu verwenden, um den Druckvorgang zu beschleunigen. Dies betrifft die Extrusionsbreite, vor allem aber die größere mögliche Schichtdicke für *Microlayering*. Wie in Abschnitt 3.1.1 beschrieben, kann das

Wechseln zwischen den einzelnen Extrudern signifikant Zeit in Anspruch nehmen. Ist die Infill-Fläche einer Ebene klein, lohnt sich der Wechsel zu einem größeren Extruder zeitlich nicht. Dies ist genau dann der Fall, wenn das Wechseln des Extruders länger dauert als die Zeitdifferenz zwischen dem Füllen mit kleiner bzw. großer Düse.
Nachdem die Perimeterbreite berechnet wurde, kann die Infill-Fläche durch Invertieren des Perimeters abgeschätzt werden. Diese Abschätzung ist nicht vollständig korrekt, da die Behandlung von Füllflächen Sonderfälle wie kleine Spalten, Reduzierung der Perimeter oder der Druckgeschwindigkeit enthält. Diese verkürzen oder verlängern die Füllzeit. Die Annäherung reicht aber für eine Entscheidung aus. **Slic3r** sieht nur einen einzigen, global zugewiesenen Extruder für Infill vor und fasst auf der Basis dessen Düsendurchmessers Infill-Ebenen zusammen, um Microlayering zu realisieren. Dabei wird das Objekt vertikal durchlaufen und so lange Ebenen zusammengefasst, wie die resultierende Schichtdicke den Extruderdurchmesser nicht übersteigt. Die Informationen über `Extruder`- und `Flow`-Objekt für den Infill-Bereich müssen nun ebenfalls pro Schicht vorliegen. Dazu werden die in Abschnitt 4.2.1 eingeführten lokalen Kopien der Objekte verwendet. Beim Zusammenfassen der Infill-Ebenen wird der Extruder der untersten Ebene jedes Durchlaufs als Referenz verwendet. Es werden so lange Ebenen zusammengefasst, bis sich der Extruderdurchmesser ändert oder die kombinierte Schichtdicke den Extruderdurchmesser übersteigt.
Die für einen Extruderwechsel benötigte Zeitspanne hängt vom verwendeten Drucker ab und muss daher vom Benutzer konfiguriert werden.

4.3. Tests

Für den Entwicklungsprozess von **Slic3r** werden, wie häufig in Softwareprojekten einer gewissen Größe, Tests eingesetzt um die Konsistenz des Quellcodes abzusichern. Für die im Rahmen dieser Arbeit implementierten Algorithmen wurden entsprechende Testobjekte geschrieben. Für den Testprozess wird jeweils ein Objekt verwendet, dass Eigenschaften aufweist, die im Algorithmus die zu prüfenden Aktionen auslösen.

Trivialere Berechnungen können durch direktes Aufrufen der entsprechenden Methoden getestet werden. Für die Berechnung von `spacing` aus der `extrusion_width` wird der Parameterraum aus Extruderdurchmesser, Schichtdicke und `spacing` iterativ in 0.2mm-Schritten durchlaufen, die Funktion für jede Kombination der Parameter aufgerufen und das Ergebnis der Berechnung mit dem bekannten korrekten Ergebnis verglichen. Analog wird die Berechnung des *Cusp*-Wertes und die Detektion horizontaler Flächen überprüft. Für das Zusammenspiel aller Komponenten werden zu Beginn jedes Tests geeignete Konfigurationsparameter gesetzt, der gesamte Slicing-Prozess durchlaufen und der resultierende G-Code analysiert. Für die Analyse

des G-Codes existiert bereits ein einfacher Parser, der eine Reihe von Werten, beispielsweise die zurückgelegte Distanz in X-Richtung, extrahiert und pro Layer zusammenfasst.

Die adaptive Berechnung der Schichtdicke wird getestet, indem überprüft wird, ob eine bestimmte Z-Koordinate angefahren wird. Mit geeignet gesetzten Parametern für die Dicke der ersten Schicht und den *Cusp*-Wert werden damit folgende Eigenschaften getestet:

- Die Reduzierung der Schichtdicke aufgrund der *Cusp*-Höhe eines stärker geneigten Facets innerhalb der Schicht
- Die Reduzierung der Schichtdicke auf die Z-Differenz eines stärker geneigten Facets, welches sich sich so hoch befindet, dass seine *Cusp*-Höhe den Layer stärker reduzieren würde als nötig.
- Das Verringern der Schichtdicke unter einer horizontalen Fläche, um eine weitere Schicht zu ermöglichen
- Das Vergrößern der Schichtdicke weil eine Verringerung nicht möglich ist

Für den Test der adaptiven Extrusionsbreite werden in einer Referenzschicht alle Linien mit der Eigenschaft $X_1 = X_2$ und $Y_1 < Y_2$ extrahiert. Damit werden nur die Perimeterdurchläufe auf einer Seite des Objekts isoliert. Infillbewegungen verlaufen diagonal, bei den Linien der anderen Seite des Objekts ist $Y_1 > Y_2$. Die Abstände der gefundenen Linien werden untersucht auf:

- Keine Reduzierung der Extrusionsbreite, das Objekt kann vollständig wiedergegeben werden
- Eine Reduzierung der Extrusionsbreite, bei der die Anzahl der Perimeter erhalten bleibt
- Eine Reduzierung der Extrusionsbreite, bei der die Anzahl der Perimeter reduziert wird. Dieser Fall entspricht auch einem Extruderwechsel.

5. Oberflächenqualität

Die Berechnung der Schichtdicken mit der *Cusp*-Methode hat einen entscheidenen Nachteil: der extrudierte Kunststoff wird als Rechteck approximiert und damit die Qualität der Oberfläche, insbesondere an vertikalen Flächen, signifikant zu hoch eingeschätzt. Die in Abschnitt 2.3.1 vorgestellten Arbeiten verwenden die mittlere Rauheit R_a zur Bewertung der Oberflächenqualität. In diesem Kapitel wird ein alternativer Ansatz über die Abweichung des Volumens vorgestellt. Es soll analysiert werden, wie sich die Oberfläche modellieren lässt, ob der Extruderdurchmesser eine Rolle spielt, wie groß die absolute Abweichung von der Solloberfläche ist und wie sich die unterschiedlichen Ansätze zur Berechnung der Schichtdicke sinnvoll kombinieren lassen.

5.1. Volumetrischer Fehler

Als volumetrischer Fehler werden mindestens zwei Phänomene bezeichnet:

- Der Fehler der im Volumen eines gedruckten Körpers durch die schichtweise Approximation entsteht (Stair-Stepping Effekt, siehe Abschnitt 2.3.1). Jeder Körper der nicht aus regulären Oberflächen besteht, gewinnt oder verliert durch das Aufteilen in Schichten an Volumen. Ein Ansatz, dieses Fehlermaß zur Steuerung der Schichthöhe zu verwenden wird beschrieben in [TFBA98, S. 158f].
- Der Fehler, der ensteht, weil die glatte Oberfläche eines Objekts aus „runden“ Formen zusammengesetzt wird (Abb. 5.1). In diesem Abschnitt geht es um die Frage, wie dieser Fehler beschrieben werden kann, von welchen Faktoren er abhängt und wie er zur Steuerung des Slicing-Prozesses eingesetzt werden kann.

Die vertikale Oberfläche eines Objekts entsteht beim FDM-Verfahren durch das Auftragen mehrerer Kunststoffschichten übereinander. Dabei „verschmiert“ der Extruder durch die horizontale Bewegung den kreisförmig extrudierten Kunststofffaden. Es entsteht vereinfacht ausgedrückt eine Schicht mit glatter horizontaler Oberfläche und Rundungen an den Seiten. Je flacher die Schicht, desto stärker wird der Kunststoff verschmiert und desto kleiner fallen die seitlichen Rundungen aus.

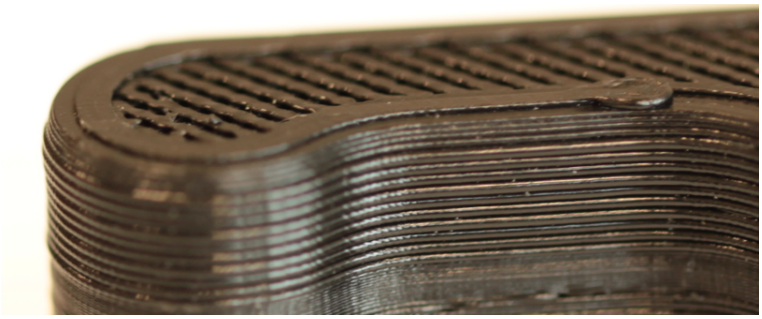


Abbildung 5.1. – Unfertiges Objekt während des Druckvorgangs. An der Oberkante ist der „verschmierte“ Kunststofffaden zu erkennen. Die vertikale Oberfläche ist zusammengesetzt aus den seitlichen Rundungen der Kunststoffschicht. In Bereichen mit geringer Schichtdicke ist die Oberfläche deutlich glatter.

5.1.1. Fragestellung

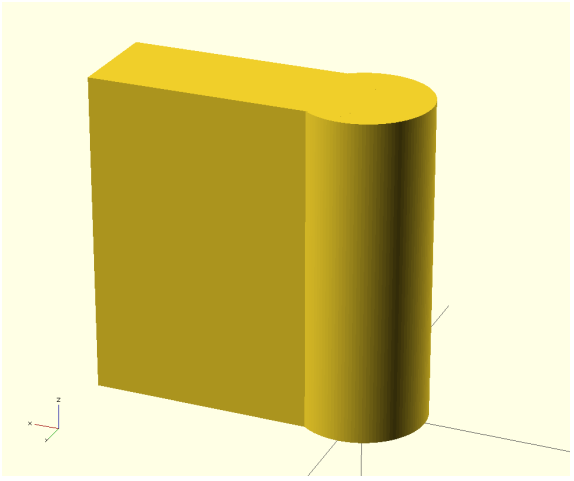
Um die Geometrie der so entstehenden Oberflächen sowohl qualitativ als auch quantitativ korrekt beschreiben zu können, wurde eine Versuchsreihe mit folgenden Fragestellungen durchgeführt:

- Mit welchen geometrischen Primitiven lässt sich die Oberfläche gut approximieren?
- Hat der Durchmesser des Extruders bei gleicher Schichthöhe Auswirkungen auf die Oberflächengeometrie?
- Wie ist das Verhältnis C_r von Schichtdicke zur Höhe einer einzelnen Wölbung der Oberfläche?

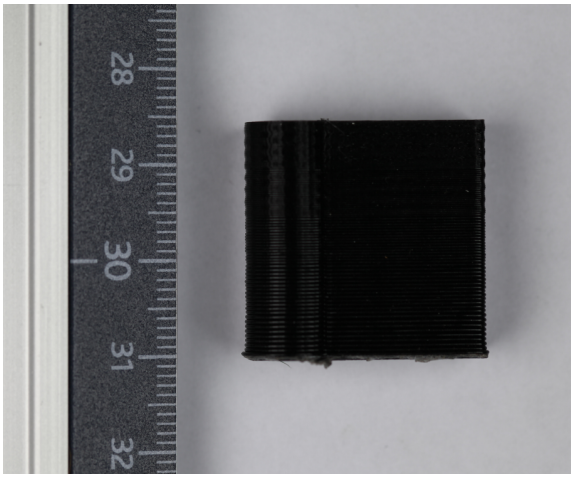
Lässt sich die gedruckte Oberfläche in Abhängigkeit von Schichtdicke und Extruderdurchmesser geometrisch beschreiben, kann das Fehlvolumen leicht als Differenz zwischen einer ebenen Oberfläche und der modellierten Oberfläche berechnet werden.

5.1.2. Durchführung

Die Oberflächen mehrerer Objekte wurden für die Vermessung mit einer optischen Kamera erfasst. Dazu wurde zunächst ein Objekt erstellt (Abbildung 5.2a), an dessen runder Seite die optische Erfassung der Oberfläche möglichst störungsfrei durchführbar ist. Das Aufschneiden eines Objektes wäre prinzipiell besser gewesen, beschädigt aber die zu erfassende Oberfläche zu stark. Das Modell wurde mit 0.5mm und 0.25mm Extrudern gedruckt. Dabei wurde mit der maximal möglichen Schichtdicke begonnen (0.5mm bzw. 0.25mm) und die Schichtdicke alle 5mm um 0.1mm verringert. (Detaillierte Abbildung in Anhang A.1). Als Druckmaterial wurde Polylactic Acid - Polymilchsäure (PLA) verwendet.



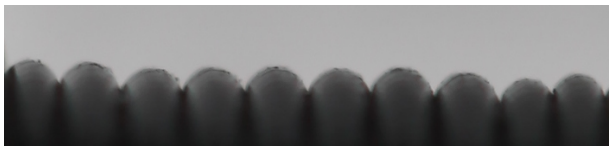
(a) CSG-Modell des zu druckenden Versuchsobjekts



(b) Gedrucktes Versuchsobjekt in der Aufnahmeumgebung

Abbildung 5.2. – Objekte zur empirischen Vermessung der Oberfläche gedruckter Gegenstände.

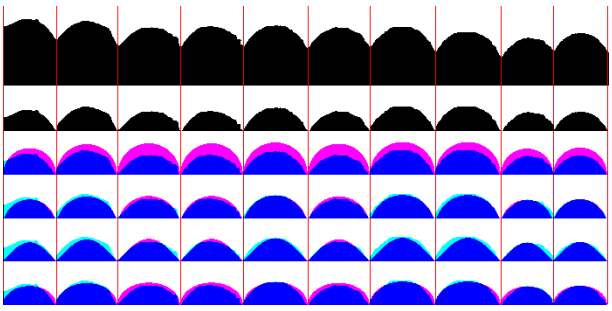
Die Kamera wurde auf einem verfahrbaren Stativ befestigt, um den Abstand zum Objekt konstant zu halten aber von jedem Bereich der Oberfläche eine lokal scharfe Aufnahme zu erhalten (Abb. 5.2b). Von jedem Objekt wurden im Abstand von 5mm Aufnahmen der Bereiche mit konstanter Schichthöhe gemacht (Abb. 5.3a). Diese wurden mit einem manuell festgelegten Schwellwert binarisiert um eine automatische Auswertung zu ermöglichen (Abb. 5.3b und 5.3c).



(a) Gedrehter Ausschnitt des Originalbildes, Schichdicke: 0.5mm

(b) Binarisierter Ausschnitt des Originalbildes [0.5mm]

(c) Binarisierter Ausschnitt des Originalbildes [0.25mm]



(d) Auswertung des vorvearbeiteten Bildes. Von oben nach unten: Originalausschnitt, zugeschnittene Erhebungen, Überlagerung mit Halbkreis, geschnittenem Halbkreis, Sinus, Halbellipse.

Abbildung 5.3. – Verarbeitungsschritte zur Auswertung der erfassten Bilddaten.

5.1.3. Auswertung

Die folgenden Arbeitsschritte sind in Abb. 5.3d dokumentiert. Jede Zeile entspricht dabei einem Arbeitsschritt.
In den vorbereiteten Rohdaten wurden, unter Anwendung von Bildverabeitungsalgorithmen, zunächst die einzelnen Erhebungen der Oberfläche isoliert (Zeile 1) und normalisiert (Zeile 2). Dabei wird jede Erhebung einzeln am untersten weißen Pixel und am obersten schwarzen Pixel abgeschnitten. Dies reduziert während des Druckprozesses induzierte mechanische Ungenauigkeiten, wie Umkehrspiel oder Vibrationen, welche eine Verschiebung der Schichten gegeneinander verursachen und erzeugt eine definierte Unterkante.
Die Zeilen 3-6 visualisieren die Überlagerung jeder Erhebung mit den unten aufgelisteten geometrischen Primitiven. Für die Konstruktion dieser sind 2 Parameter nötig: die Breite oder Periode P der Erhebungen und ihre Amplitude A. Die Periode entspricht der Schichtdicke des Objekts. Sie wird aus dem Bild als der Mittelwert der Abstände zwischen zwischen den Minima (rote Linien) errechnet. Die Amplitude kann aus dem Bild als Mittelwert der Maxima errechnet werden. Das Verhältnis von Periode zu Amplitude $C_r = \frac{P}{A}$ ist die in der Fragestellung gesuchte Variable. Folgende geometrische Primitive wurden als Kandidaten für die Approximation der Oberfläche verwendet:

Halbkreis Ein Halbkreis mit Durchmesser P: $\sqrt{r^2 - (Px - r)^2}$ mit $r = \frac{P}{2}$, $x \in [0, 1]$. Die Amplitude des Halbkreises ist folglich $\frac{P}{2}$.

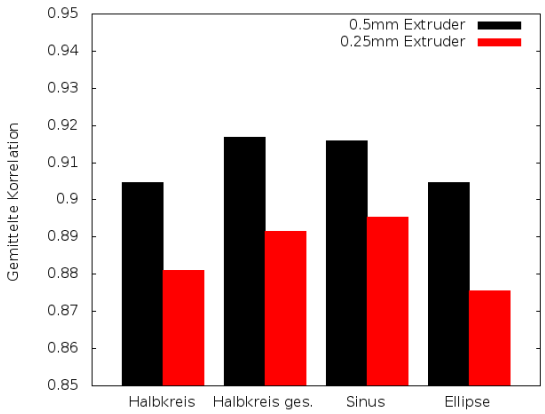
Geschnittener Halbkreis Um die Amplitude besser zu approximieren, wird der untere Teil bei A abgeschnitten.

Sinus Sinusfunktion im Bereich $[0, \pi]$: $\sin(x \cdot \frac{\pi}{P}) \cdot A$.

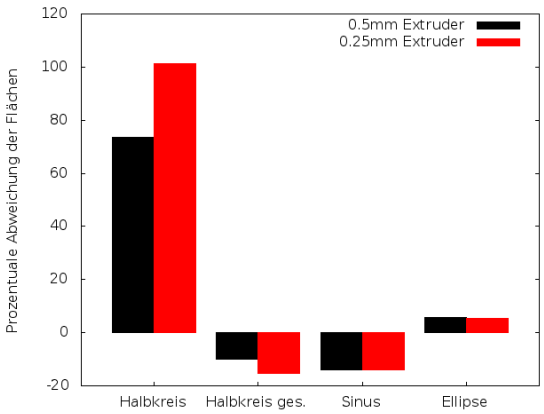
Ellipse Halbellipse, erzeugt aus einem mit $\frac{2}{C_r}$ skalierten Halbkreis.

Zur Bewertung der geometrischen Primitive wurden jeweils die Korrelation (Abb. 5.4a) und die Integraldifferenz[1] (Abb. 5.4b) zwischen erzeugter und gemessener Oberfläche berechnet.
Die Korrelationswerte aller vier Primitive liegen in der gleichen Größenordnung. Betrachtet man die Flächenintegrale und somit das Volumen der Oberfläche, zeigt sich, dass die Erhebungen am ehesten abgeflachten Halbkreisen entsprechen und entsprechend gut durch Halbellipsen approximiert werden können.

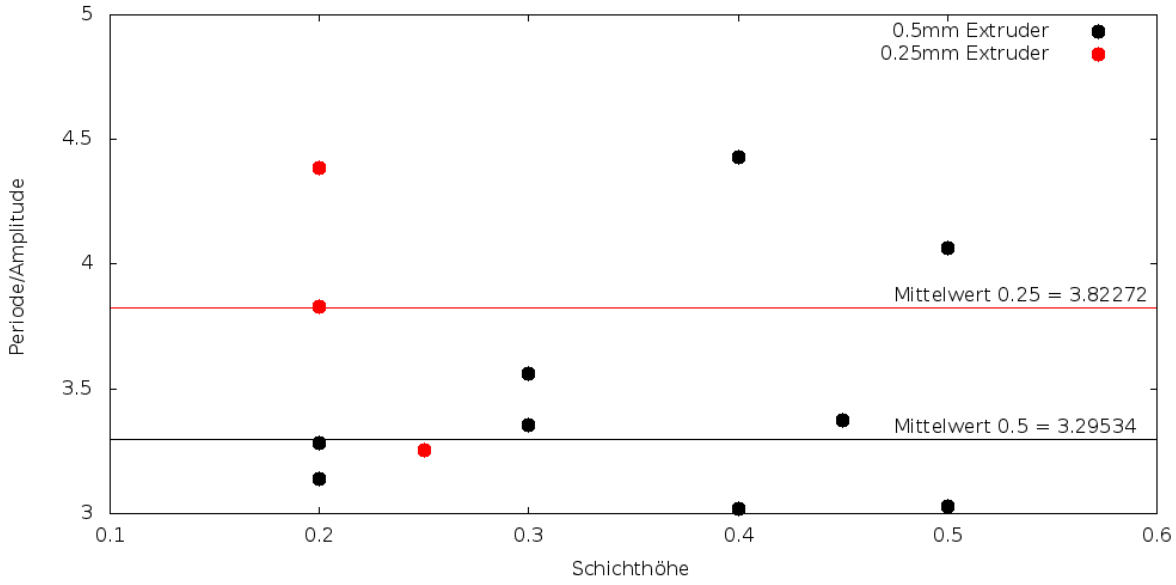
[1] Eine detallierte Visualisierung der Datenpunkte findet sich im Anhang, Abb. A.2



(a) Korrelation zwischen gemessener und konstruierter Oberfläche, berechnet für jede einzelne Erhebung und anschließend über alle Messpunkte einer Schichtdicke pro Objekt gemittelt.



(b) Prozentuale Abweichung des Integrals der konstruierten Oberfläche von der gemessenen Oberfläche.



(c) Verhältnis C_r von gemessener Schichtdicke P zur Höhe der einzelnen Erhebungen A gemittelt über alle Messpunkte einer Schichtdicke pro Objekt.

Abbildung 5.4. – Auswertung der Messdaten zur Oberflächengeometrie.

Abbildung 5.4c zeigt die Verteilung aller gemessenen C_r. Die hohe Variation ist bedingt durch die beim FDM-Verfahren, insbesondere bei Verwendung kostengünstiger Hardware, insgesamt niedrige Homogenität der erzeugten Oberflächen. Die gemessenen Werte sind weder von der Schichthöhe noch vom Extruderdurchmesser signifikant abhängig. Die geringe Anzahl der Messpunkte lässt darüber keine sichere Beurteilung zu, Schwankungen in der gemessenen Größenordnung spielen aber für die gesuchte Bewertung der Oberfläche nur eine untergeordnete Rolle. In der Praxis hat sich ein Wert von $C_r = 3.5$ als guter Mittelwert erwiesen. Dieses Ergebnis deckt sich mit dem von Pandey et al. ermittelten Wert von $C_r \approx 3.3$ [PRD03b, S. 64].

Der volumetrische Fehler in der Oberfläche lässt sich somit berechnen aus dem Volumen eines elliptischen Halbzylinders V_c:

$$V_c = \frac{1}{2} \cdot r \cdot \frac{r}{C_r} \cdot \pi \cdot 1 = \frac{r^2\pi}{2C_r} \tag{5.1}$$

und dem Volumen V_s, welches sich durch die Solloberfläche ergeben würde:

$$V_s = 2r \cdot \frac{r}{C_r} \cdot 1 = \frac{2r^2}{C_r} \tag{5.2}$$

Normiert über die Oberfläche ($2 \cdot r \cdot 1 = 2r$), ergibt sich das Fehlvolumen V_e:

$$V_e = \frac{V_s - V_c}{2r} = \frac{\frac{2r^2}{C_r} - \frac{r^2\pi}{2C_r}}{2r} = 4 - \pi \cdot \frac{r}{C_r} \approx 0.2146 \cdot \frac{r}{C_r}\ \mathrm{mm}^3/\mathrm{mm}^2 \text{ wobei } r = \frac{P}{2} \tag{5.3}$$

Unter dem Aspekt der Benutzerfreundlichkeit macht es möglicherweise mehr Sinn, den Fehler in $\mathrm{mm}^3/\mathrm{cm}^2$ anzugeben um intuitiv verständlichere Werte zu erhalten.

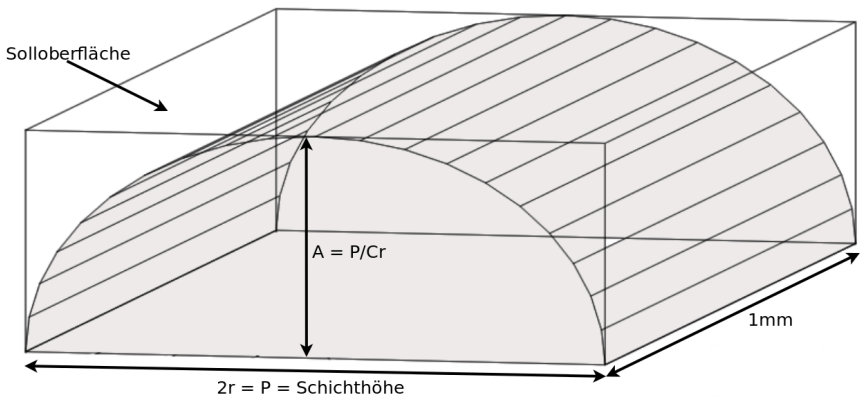


Abbildung 5.5. – Volumetrischer Fehler der Oberfläche: die Differenz zwischen Quader (Solloberfläche) und eingefärbtem Zylinder (tatsächliche Oberfläche).

5.1.4. Ergebnis

Der volumetrische Fehler der Oberfläche hängt hauptsächlich von der Schichtdicke ab und lässt sich aus dieser leicht errechnen.

- Die Oberfläche lässt sich mit 3 der 4 untersuchten geometrischen Primitive prinzipiell ausreichend gut approximieren. Halbellipsen liefern mit den erhobenen Daten die besten Ergebnisse und der Fehler lässt sich so in Abhängigkeit von der Schichtdicke linear beschreiben.
- Der Durchmesser des Extruders hat keinen signifikanten Einfluss auf die Qualität der Oberfläche, sie hängt wesentlich nur von der Schichtdicke und anderen mechanischen Eigenschaften des Druckers und des Materials ab.
- Das Verhältnis von Schichthöhe zur Höhe einer einzelnen Wölbung der Oberfläche ist konstant. Die gesuchte Konstante C_r wurde mit ausreichender Genauigkeit auf 3.5 bestimmt.

5.2. Integration der Verfahren

Es wurden im Verlauf der Arbeit eine Reihe von Parametern identifiziert, die Einfluss auf den Slicing-Prozess haben und vom Benutzer potentiell variiert werden können:

- Minimal/maximal mögliche Schichthöhe pro Extruder
- Zuordnung der Extruder
- Oberflächenfehler V_e
- Cusp Wert C_{max}
- Fehler in spitzen Winkeln E
- Extrusionsfaktor f_e
- Fehlerschwellwert für binäre Suche ε

Diese Optionen können in der existierenden Benutzeroberfläche dort integriert werden, wo sie Alternativen zu bereits existierenden Optionen sind. So kann der *Cusp* Wert etwa bei den Einstellungen zur Schichtdicke platziert werden. Da alle Implementierungen bislang als experimentell betrachtet werden können, wurden die Optionen soweit wie möglich in einem zentralen Tab zusammengefasst. Eine Ausnahme bilden hier die Begrenzungen der Schichthöhe. Da sie Eigenschaften der separat modellierten Extruder sind, wurden die Optionen dort untergebracht. Um die Komplexität für Anwender geringer zu halten wurden, die Parameter E, f_e und ε statisch auf die in den jeweiligen Abschnitten begründeten Werte gesetzt.

Die Wahl der Schichtdicke ist grundlegend ein Kompromiss zwischen Qualität und Geschwindigkeit. Da der Volumenfehler der Oberfläche V_e direkt linear mit der Schichtdicke zusammenhängt, kann direkt ein Wert zwischen maximaler Qualität (L_{min}) und maximaler Geschwindigkeit (L_{max}) gewählt werden. Der *Cusp* Wert tendiert ebenfalls dazu, bei hoher Präzision kleinere Schichtdicken und somit niedrigere Geschwindigkeiten zu erzeugen, verhält sich aber konzeptionell bei annähernd vertikalen Flächen anders. Präzision und Oberflächenqualität korrelieren also bei großen Winkeln, divergieren aber mit abnehmendem Winkel, wenn die Präzision größer wird, die Oberflächenqualität aber konstant bleibt.

Um diesen Sachverhalt abzubilden, wurde folgender Ansatz verwendet:
Es wird zunächst das Verhältnis Geschwindigkeit $\leftrightarrow$ Oberflächenqualität V_e mit einem Wert zwischen 0 und 1 festgelegt. Es wird ein *Cusp* Wert C_{max} festgelegt und anschließend eine Gewichtung zwischen Oberflächenfehler und Präzisionsfehler ω (0-1) gewählt. Dabei wird C_{max} wie in Abschnitt 4.1.1 begründet, auf das Intervall [0, L_{max}] begrenzt. Nach der Berechnung einer Schichtdicke durch den *Cusp*-Algorithmus wird ein mit ω gewichtetes Mittel aus Schichtdicke nach Oberflächenfehler und Schichtdicke nach Präzisionsfehler gebildet. Durch die Bildung des Mittelwertes kann es vorkommen, dass horizontale Oberflächen nicht mehr korrekt abgebildet werden. Um dies zu vermeiden, wird der Abstand zur nächsten horizontalen Oberfläche auf Basis des Präzisionsfehlers ermittelt. Wurde keine horizontale Fläche gefunden, kann der Mittelwert verwendet werden. Wurde eine horizontale Fläche gefunden, wird die Schichtdicke, wenn möglich, wie in Abschnitt 4.1.2 beschrieben, soweit verringert, dass unterhalb der Oberfläche eine weitere Schicht realisiert werden kann. Ist der Mittelwert noch geringer, wird sie auf Diesen reduziert. Ist eine Reduzierung nicht möglich, wird die Schicht bis zur horizontalen Fläche erweitert.

Um die wenig intuitive Konfiguration für Anwender weiter zu vereinfachen, lassen sich die Parameter V_e und C_{max} koppeln. Für C_{max} wird dann der Wertebereich $[L_{min}, L_{max}]$ verwendet und das Intervall $[0, 1]$ abgebildet. Die gesamte Steuerung der Schichtdicke erfolgt dann über einen einzigen Parameter, der die Gewichtung zwischen Geschwindigkeit und Qualität regelt, und eine (optionale) Gewichtung zwischen den Verfahren.

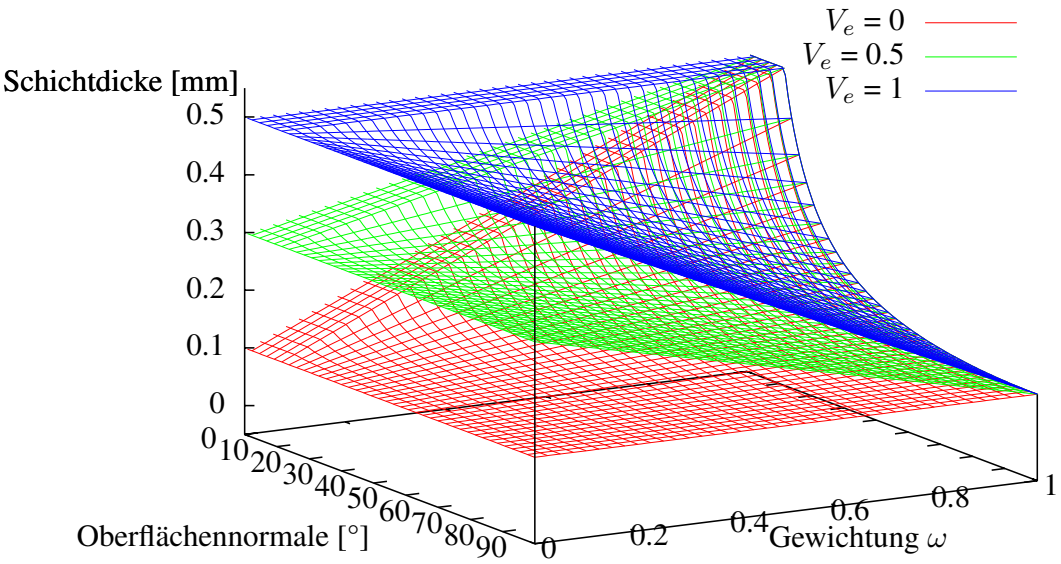


Abbildung 5.6. – Durch Oberflächenqualität und Präzision aufgespannter Parameterraum. Links wird die Schichtdicke nur nach V_e berechnet, rechts nach C_{max}. Die Parameter in dieser Abbildung sind: $L_{min} = 0.1$, $L_{max} = 0.5$, $C_{max} = 0.1$.

6. Evaluation

Nach erfolgter Implementierung werden die Resultate in diesem Abschnitt auf ihre Funktionalität und Effizienz untersucht. Der Fokus liegt dabei auf den Verbesserungen, die bei der Druckqualität und Geschwindigkeit erzielt werden konnten. Es wird eine kurze theoretische Abschätzung angegeben, sowie eine empirische Auswertung tatsächlicher Druckvorgänge durchgeführt.

6.1. Druckergebnisse

Die Verbesserung der Druckergebnisse lässt sich grob aufteilen in eine Verbesserung der Qualität und die Verringerung der Druckzeit. Diese beiden Eigenschaften verhalten sich grundsätzlich diametral. Dabei ist der Effekt aber nicht in jedem Fall ein Kompromiss zwischen diesen beiden Zielparametern. Die Berücksichtigung horizontaler Flächen etwa, hat zwar eine Auswirkung auf einen bestimmten Qualitätsaspekt, die Veränderung der Druckzeit ist aber nur marginal. Die Auswirkungen sind zudem in hohem Maße abhängig von der Geometrie der verarbeiteten Objekte.

6.1.1. Qualität

Horizontale Flächen

Horizontale Flächen wurden bislang in **Slic3r** nicht gesondert berücksichtigt und daher durch die nächstliegende Ebene abgebildet. Ihre maximale Abweichung beträgt daher $\frac{Schichtdicke}{2}$, der mittlere Fehler $\frac{Schichtdicke}{4}$. Bei adaptivem Slicing beträgt die Abweichung 0, wenn der Abstand zweier Flächen kleiner ist als die minimale Schichtdicke.

Es wurde ein Testobjekt mit zufällig generierten Höhenstufen (Abb. 6.1) gedruckt und die Abweichung der Flächen gemessen. Jede Konfiguration wurde zweimal gedruckt und die Ergebnisse gemittelt. Die gemessenen Abweichungen waren dabei nahezu identisch. Bei adaptiver Schichtdicke ist die Abweichung erwartungsgemäß am kleinsten, bei gleichzeitig niedriger Druckzeit. Bei statischer 0.1mm-Schichtdicke ist die Abweichung etwas größer, bei dafür sehr viel höherer Druckzeit. Eine erstaunlich niedrige Abweichung ergibt sich bei statischer

Experiment	0.4mm static	0.1mm static	adaptive	0.1mm static (microlayers)
Druckzeit [mm:ss]	08:50	27:28	10:40	22:15
Mittl. Abweichung [mm]	0.076	0.045	0.039	0.108
Schichten	16	63	20	63
Relation (Abweichung)	51	87	100	36

Tabelle 6.1. – Abweichung horizontaler Flächen bei verschiedenen Slicing-Verfahren. Feste Schichtdicke (static), variable Schichtdicke (adaptive) und feste Schichtdicke mit kombiniertem Infill (microlayers). Die Zeile Relation beschreibt die Abweichung im Verhältnis zur geringsten gemessenen Abweichung um eine direkte Vergleichbarkeit zu ermöglichen.

0.4mm-Schichtdicke. Dies hängt vermutlich damit zusammen, dass die Zufallswerte des Modells annähernd Vielfachen von 0.4 entsprechen. Statische 0.1mm-Schichtdicke mit kombinierten Infill-Layern erzeugt eine verhältnismäßig hohe Druckzeit bei erstaunlich niedriger Genauigkeit. Vermutlich wird dieser Effekt durch die stark variierende Dicke des Infills erzeugt.
Die Berücksichtigung horizontaler Flächen wird erst durch variable Schichtdicken ermöglicht und führt zu einem signifikanten Anstieg der Genauigkeit.

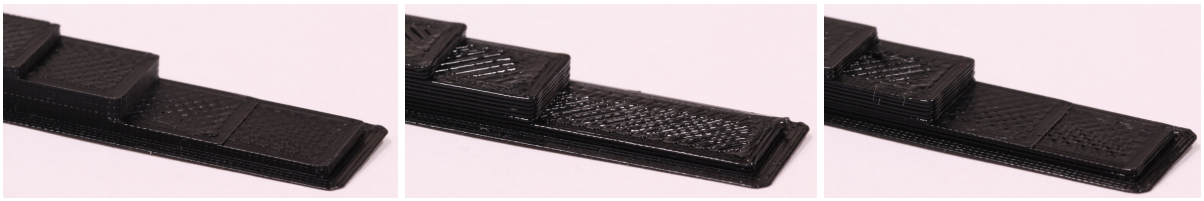


Abbildung 6.1. – Testobjekte mit zufällig verteilten Höhenstufen zur Bewertung der Präzision horizontaler Oberflächen. Auflösung von Links nach Rechts: 0.1mm, 0.4mm, adaptiv.

Spitze Winkel

Eine Abschätzung für die Genauigkeit spitzer Winkel in Abhängigkeit vom Extruderradius wurde bereits in Abschnitt 2.3.2 und in Abb. 4.8 angegeben. Der Effekt für unterschiedliche Extruderradien und für die Verringerung der Extrusionsbreite für einen Extruder wurde zusätzlich experimentell überprüft. Dazu wurde ein 35° Winkel mit einem 0.5mm und einem 0.25mm Extruder gedruckt. Dabei wurden Extrusionsbreiten von 0.25, 0.5 und 0.75mm verwendet. Nur der 0.25mm Druck wurde mit dem kleinen Extruder als Referenzdruck durchgeführt. Zusätzlich wurde ein Objekt mit 0.5mm Extruder und adaptiver Extrusionsbreite gedruckt, bei 35° ergibt das eine Extrusionsbreite von 0.345mm (vgl. Abb. 4.8).
Die erzeugten Objekte wurden anschließend optisch mit einer Kamera erfasst um die kleinen Streckendifferenzen mit hinreichender Präzision zu messen. Die Ergebnisse sind in Tabelle 6.2 dargestellt.

Extrusionsbreite	0.25mm	0.345mm (dynamic)	0.5mm	0.75mm
Fehler erwartet	0.291mm	0.401mm	0.581mm	0.872mm
Fehler gemessen	**0.787mm**	**1.090mm**	**1.271mm**	**1.412mm**

Tabelle 6.2. – Fehler in spitzen Winkeln, abhängig vom Durchmesser des Extruders. Der erwartete Fehler ist berechnet nach der Gleichung in Abb. 2.6 (Vgl. [BRA$^+$]).

Es zeigt sich, dass sowohl der Wechsel zu einem kleineren Extruder als auch das Skalieren der Extrusionsbreite einen signifikanten Einfluss auf den resultierenden Fehler haben. Subjektiv ist die Verbesserung für einen menschlichen Betrachter ohne Hilfsmittel nicht oder nur schwach erkennbar.

6.1.2. Druckzeit

Stair-Stepping

Der Stair-Stepping-Effekt ist eines der zentralen adressierten Probleme. Eine Abschätzung der Verkürzung der Druckzeit für einen gegebenen Fehler C_{max} durch adaptives Slicing ist wegen der Aufteilung des Objekts in Infill und Perimeter nicht trivial darstellbar. Bei statischem Slicing setzt sich die Druckzeit t_s zusammen aus der benötigten Zeit für Infill und Perimeter:

$$t_s = t_i + t_p \tag{6.1}$$

Das Generieren von Infill ist pro Volumen um einen Faktor φ schneller als das Generieren von Perimeter. Die Aufteilung der gesamten Druckzeit t_s auf Infill und Perimeter ist daher:

$$t_s = t_s \cdot \frac{V_i \cdot \varphi}{V_i \cdot \varphi + V_p} + t_s \cdot \frac{V_p}{V_i \cdot \varphi + V_p} \tag{6.2}$$

Wobei V_i das Infillvolumen, V_p das Perimetervolumen und V_o das Objektvolumen sind.
Die Verteilung der Volumina entspricht näherungsweise dem Verhältnis von Volumen V_o zu Oberfläche S des Objekts. Für das Volumen des Perimeters V_p gilt (näherungsweise):

$$V_p \approx S \cdot p \tag{6.3}$$

wobei p die Perimeterbreite ist und im Folgenden angenommen wird das $p = 1$mm. Mit dieser Vereinfachung gilt $V_p = S$, wenn die Einheiten vernachlässigt werden. Das Volumen des Infills V_i ist damit:

$$V_i = V_o - V_p = V_o - S \tag{6.4}$$

Die jeweiligen Anteile am Gesamtvolumen sind $\beta = \frac{S}{V_o}$ bzw. $1 - \beta = \frac{V_o - S}{V_o}$ und entsprechen somit genau dem Verhältnis zwischen Oberfläche und Volumen. Diese Abschätzung wird unge-

nauer für Objekte mit großer Oberfläche im Verhältnis zum Volumen und ist nur gültig, solange der mm^2-Wert Oberfläche kleiner ist als der mm^3-Wert des Volumens. In diesem Grenzbereich überwiegt der Perimeteranteil aber den Infillanteil bei realen Objekten ohnehin so stark, dass ein Grenzfaktor von 0 ausreichend genau ist.
Für die Verteilung der Druckzeit auf Infill und Perimeter gilt nun bei statischem Slicing:

$$t_s = t_s \cdot \frac{(1-\beta)\cdot\varphi}{(1-\beta)\cdot\varphi+\beta} + t_s \cdot \frac{\beta}{(1-\beta)\cdot\varphi+\beta} \tag{6.5}$$

Bei dynamischem Slicing hängt die Veränderung der Druckzeit davon ab, in welchem Maße Microlayering eingesetzt wurde. Verändere sich die Anzahl der für die Einhaltung von C_{max} nötigen Schichten um den Faktor α. Wird kein Microlayering eingesetzt, wirkt sich α sowohl auf Infill als auch Perimeter aus. Die Veränderung der adaptiven Druckzeit t_a gegenüber t_s ist dann linear:

$$t_a = \frac{t_s}{\alpha} \tag{6.6}$$

Bei Einsatz von Microlayering kann das Infillvolumen im besten Fall unverändert gedruckt werden, da ohnehin immer die maximale Schichtdicke angewendet wird. Die Verringerung der Schichtenzahl um α hat nur Auswirkungen auf das Perimetervolumen. Für die Druckzeit t_a gilt dann:

$$t_s = t_s \cdot \frac{(1-\beta)\cdot\varphi}{(1-\beta)\cdot\varphi+\beta} + \frac{t_s}{\alpha} \cdot \frac{\beta}{(1-\beta)\cdot\varphi+\beta} \tag{6.7}$$

Da Perimeter nur an vertikalen Flächen auftreten, müssten horizontale Flächen eigentlich aus der Berechnung von β ausgenommen werden. An horizontalen Flächen kommt aber microlayering nicht zum Tragen, da dort n Schichten vollständig gefüllt werden, um eine geschlossene Oberfläche zu erzeugen. Die Vereinfachte Behandlung aller Oberflächen führt daher nicht zu signifikanten Abweichungen.

Das Verhältnis φ der Druckzeit pro Volumen zwischen Infill und Perimeter hängt im Wesentlichen ab von:

- Microlayering-Faktor f_m [2]
- Füllgrad f_f [0.5]
- Druckgeschwindigkeits-Faktor f_s [1.5]

Die Werte in Klammern sind Abschätzungen für realistische Praxiswerte, können aber tatsächlich stark variieren. Einen cm^3 Infill zu erzeugen dauert also etwa φ mal so lange, wie einen cm^3 Perimeter zu erzeugen, wobei

$$\varphi = \frac{f_f}{f_m \cdot f_s} \tag{6.8}$$

Mit den angenommenen Praxiswerten gilt somit $\varphi = 0.17$.

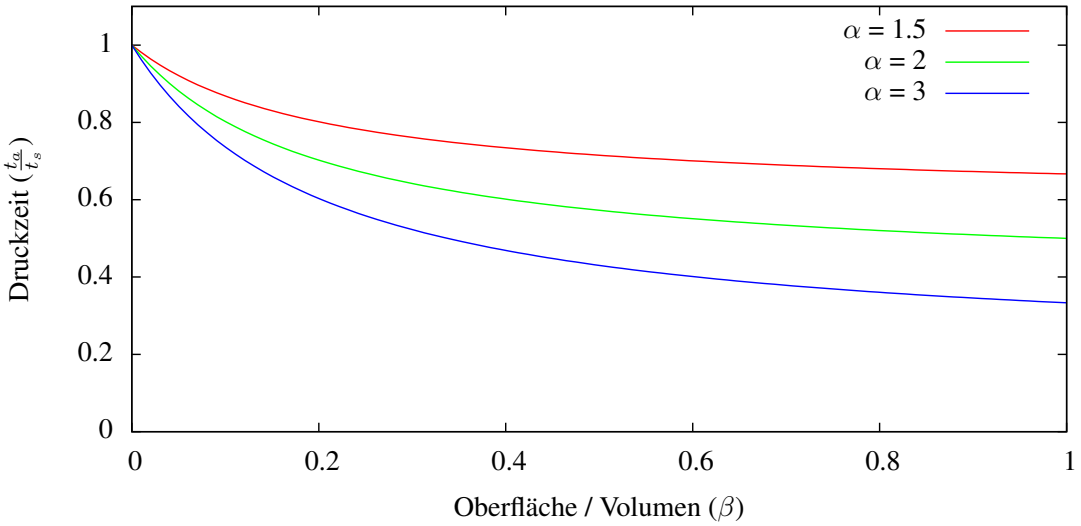


Abbildung 6.2. – Verringerung der Druckzeit t_a bei adaptivem Slicing im Verhältnis zur Druckzeit t_s bei statischem Slicing und aktivem Microlayering. Durch adaptives Slicing verringert sich die Anzahl benötigter Layer im Perimeter um den Faktor α.

Für die empirische Evaluation der Druckzeit wurde eine Reihe von Testobjekten erstellt und ausgedruckt. Um Vergleichbarkeit zu ermöglichen, sollten die Testobjekte ein möglichst konstantes α bei variierendem β aufweisen. Die für den Druck verwendeten Parameter sind: $f_m = 3$, $f_f = 0.5$, $f_s = 1.8$. Somit ist $\varphi = \frac{0.5}{3.0 \cdot 1.8} = 0.093$. Die Objekte wurden jeweils einmal mit statischem und adaptivem Slicing gedruckt. Objektbedingt ist α nicht exakt konstant, für bessere Vergleichbarkeit wurde der Unterschied in Abb. 6.4 aus den Daten herausgerechnet.

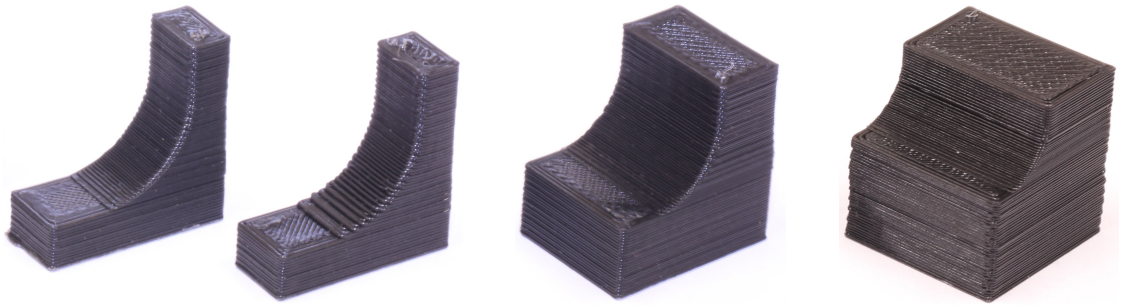


Abbildung 6.3. – Testobjekte zur empirischen Ermittlung der Verringerung der Druckzeit, von links nach rechts mit zunehmendem β (10mm - 25mm). Das 10mm-Objekt ist doppelt abgebildet, links mit adaptivem Slicing, rechts mit statischem Slicing [0.4mm].

Die Ergebnisse in Tabelle 6.3 zeigen, dass der Druckvorgang durch adaptives Slicing signifikant beschleunigt werden kann. Die benötigte Zeit konnte im besten Fall auf etwa ein Drittel reduziert werden. Dieser Zeitvorteil geht hauptsächlich zu Lasten der Oberflächenrauigkeit an vertikalen Flächen, und kommt somit insbesondere dort zum Tragen, wo Maßhaltigkeit relevan-

Experiment	10mm	20mm	25mm
Oberfläche / Volumen (β)	0.52	0.33	0.26
Druckzeit statisch (t_s)	45:34	58:04	01:12:03
Druckzeit adaptiv (t_a)	16:00	23:16	30:29
Verhältnis Schichten (α)	3.074	3.15	3.32
Zeitverhältnis	**0.356**	**0.42**	**0.456**

Tabelle 6.3. – Beschleunigung des Druckvorgangs durch adaptives Slicing in Abhängigkeit vom Verhältnis Oberfläche zu Volumen des Objekts (β).

ter ist als optische Qualität. Etwa bei der Entwicklung technischer Bauteile.
Die theoretische Abschätzung der Veränderung der Druckzeit wurde empirisch belegt. Abbildung 6.4 zeigt die Übereinstimmung der erhobenen Daten mit den vorhergesagten Werten. Dies ermöglicht es, den Effekt der Anwendung vor dem Druck abzuschätzen.
Da die Testobjekte mit gezielt gewählten Eigenschaften erzeugt wurden (konstantes α, variables β), wurde zusätzlich ein Vergleichsobjekt mit realer technischer Anwendung gedruckt: ein Gehäuseteil des in Entwicklung befindlichen druckbaren modularen Roboters [KWHZ14]. Die Druckzeit ist in Abb. 6.4 wiedergegeben (roter Punkt) und entspricht in etwa der Vorhersage. Die Oberflächenqualität ist bei einem Bauteil dieser Art weniger relevant als die Präzision, da mechanische Komponenten (Servo, Magnete, etc.) präzise in die Aufnahmen passen müssen. Während der Entwicklung des Roboters wurde eine größere Anzahl der Module gedruckt, die Zeitersparnis durch adaptives Slicing war subjektiv signifikant spürbar.

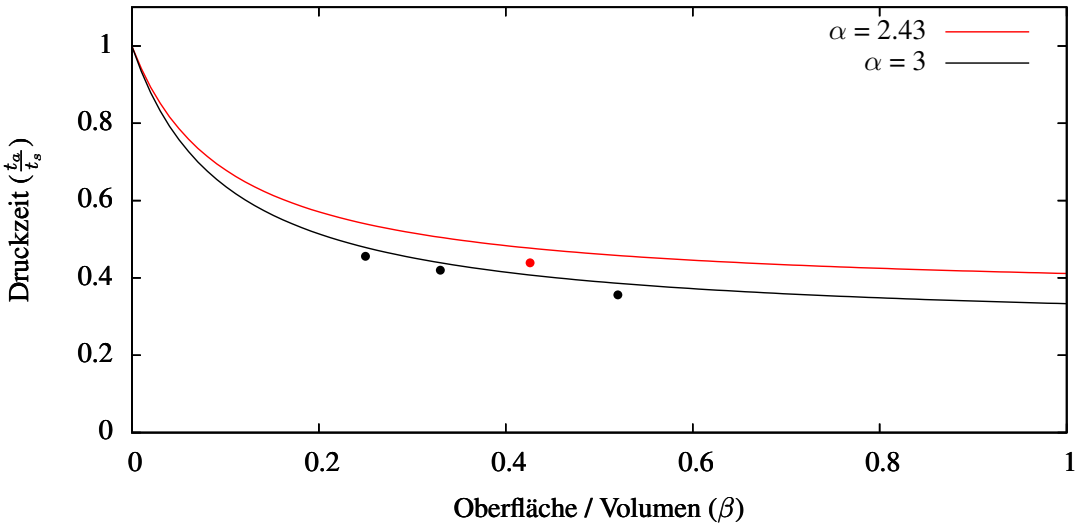


Abbildung 6.4. – Relative Druckzeit der Versuchsobjekte im Vergleich mit der Abschätzung in Gleichung (6.7) (schwarz) und relative Druckzeit eines Roboterteils als Vergleichsobjek (rot).

Extruderwechsel

Die in Abschnitt 4.2 beschriebenen Techniken zur Anpassung der Extrusionsbreite haben primär das Ziel, die Qualität der gedruckten Objekte zu verbessern. Um die Auswirkungen auf die Druckgeschwindigkeit zu evaluieren, wurde wurde ein Testobjekt mit verschiedenen Parametern gedruckt. Das Objekt ist so gestaltet, dass die Wandstärke nach oben hin abnimmt, bis eine Wandstärke von 0 erreicht ist, um den Einsatz eines möglichst kleinen Extruders zu erzwingen. Im unteren Bereich enthält das Objekt größere füllbare Flächen und einen Steg, der ebenfalls den Einsatz eines kleinen Extruders erzwingt.

Versuch Nr.	Extruder	Infill-Schwellwert	Druckzeit
1	0.25	-	41:14
2	0.25 + 0.5	5s	48:03
3	0.25 + 0.5 ohne Reinigung	5s	37:15
4	0.25 + 0.5	20s	40:28

Tabelle 6.4. – Druckzeiten eines Testobjekts mit verschiedenen Extruderkonfigurationen. Der Infill-Schwellwert gibt die geschätzte Zeit für das Füllen eines Layers an, ab der sich der Wechsel auf einen größeren Extruder lohnt (Beschrieben in Abschnitt 4.2.2).

Die Druckzeiten sind in Tabelle 6.4, Bilder der gedruckten Objekte in Abb. 6.5 dargestellt. Versuch 1 ist der Referenzversuch, es wurde nur der kleine Extruder verwendet. In den Versuchen 2-4 wurde in Bereichen, in denen es möglich ist, mit dem großen Durchmesser gedruckt. Dabei zeigt sich, dass es wesentlich darauf ankommt, wie lange der Wechsel von einem Extruder auf den anderen braucht. Der Infill-Schwellwert wurde so gewählt, dass in den Versuchen 2 und 3 in den unteren Schichten zum Füllen auf den großen Extruder gewechselt wurde. In Versuch 2 erhöht dies die Druckzeit so dramatisch, dass der Einsatz des 2. Extruders nicht lohnt. In Versuch 3 wurde das Reinigen der Extruder beim Wechsel ausgeschaltet. Die Druckzeit ist deutlich reduziert, die resultierende Qualität aber nicht akzeptabel. In Versuch 4 wurde ein realistischerer Schwellwert eingestellt. Dies führt dazu, dass in keiner Schicht der große Extruder für Infill eingesetzt wird, die Druckzeit liegt leicht unterhalb der von Versuch 1.

Die Ergebnisse zeigen, dass der Einsatz mehrerer Extruder für lokale Qualitätsstufen funktioniert und bei korrekter Konfiguration die Druckzeit bei konstanter Qualität verringert. Die Nützlichkeit hängt dabei stark von den Eigenschaften des Objekts ab. Der Effekt wäre mit tropfärmeren Extrudern deutlich größer.

Abbildung 6.5. – Testdrucke für den Einsatz zweier Extruder in Bereichen mit unterschiedlichen Anforderungen an die Auflösung. Um den Effekt sichtbar zu machen wurde im 0.25mm-Extruder rotes, im 0.5mm-Extruder schwarzes Filament verwendet. **Oben**: von Links nach Rechts Versuch 1-4. **Unten**: Links Objekt 2, im unteren Bereich wurde mit niedriger Auflösung gefüllt, Rechts Objekt 4, im unteren Bereich mit hoher Auflösung gefüllt.

6.2. Software

6.2.1. Performance

In diesem Abschnitt wird die Performance der implementierten Algorithmen auf Softwareseite untersucht. Das Slicen komplexer Modelle kann zeitintensiv sein, daher ist es relevant wie sich die Laufzeit durch die zusätzliche Analyse eines Modells verändert. Ein wesentliches Ziel adaptiver Slicing-Algorithmen ist die Verringerung der Druckzeit, eine aufwendige Analyse lohnt sich daher vor allem dann, wenn die Druckzeit der dominierende Zeitfaktor ist.

Für die Analyse der Laufzeit wurde der G-Code für 3 Modelle mit ansteigender Komplexität berechnet (Abb. 6.6).
Ein direkter Vergleich zwischen statischem und adaptivem Slicing ist nicht ohne Weiteres möglich, da die Anzahl der Schichten durch die Analyse i.A. signifikant reduziert wird. Der dominierende Teil der Rechenzeit ergibt sich durch die Pfadgenerierung nach dem eigentlichen Slicing-Prozess. Jeder Testdurchlauf wurde deshalb einmal mit statischem Slicing und einer Schichtdicke von 0.1mm durchgeführt. Anschließend mit adaptiver Berechnung der Schichtdicke und ein weiteres Mal statisch, mit einer Schichtdicke, die die selbe Anzahl Schichten generiert, wie der adaptive Prozess. Auf diese Weise ist sowohl ein direkter Vergleich zwischen statischer und adaptiver Laufzeit als auch zwischen den Laufzeiten pro Schicht möglich.
Die Veränderung der Extrusionsbreite ist weitgehend unabhängig von der Anzahl der Schichten,

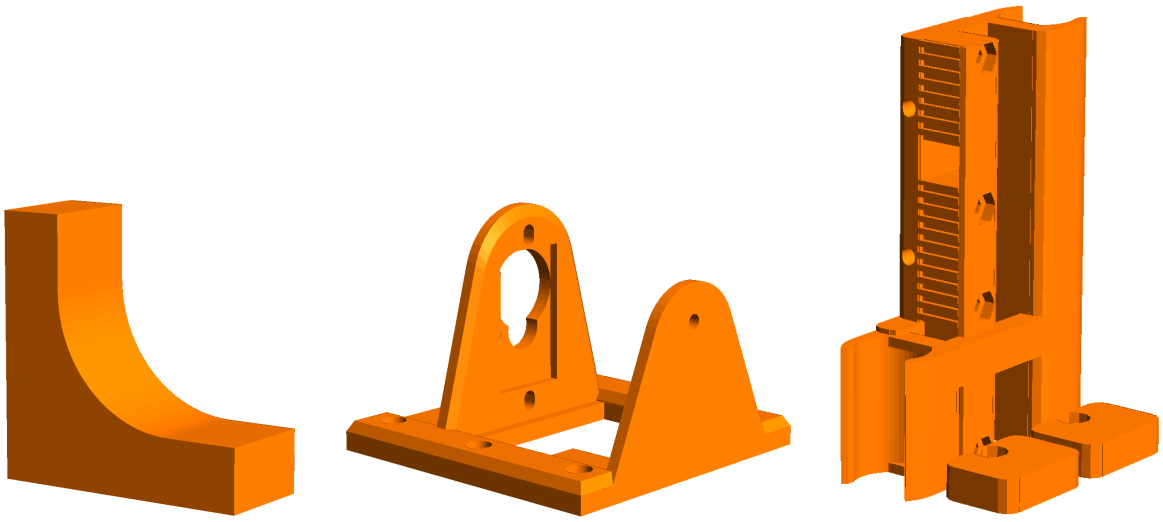


Abbildung 6.6. – Modelle zum testen der Performance, von links nach rechts in aufsteigender Komplexität: Testobjekt L-Profil, Servo-Modul des modularen Roboters [KWHZ14], X-Carriage des Prusa Mendel I3 Druckers.

da sie auf Schichtebene geschieht. Um die Laufzeit von Adaptiver Schichthöhe und Extrusionsbreite getrennt betrachten zu können, wurde jedes Modell einmal mit adaptiver Schichthöhe, einmal mit adaptiver Extrusionsbreite und einmal mit beiden Verfahren berechnet.

Es wurde eine Druckerkonfiguration mit einem 0.5mm Extruder und einem zweiten 0.25mm Extruder für adaptive Extrusionsbreite verwendet. Der *Cusp*-Wert C_{max} ist durchgängig 0.1, um für statischen und dynamischen Druck die gleiche Präzision zu erzeugen. Es wurde ausschließlich der Konturfehler berücksichtigt ($\omega = 0$), der Oberflächenfehler geht in die Berechnungsgeschwindigkeit nur über die Schichtdicke ein und verändert somit nur die Anzahl der erzeugten Schichten. Alle Berechnungen wurden in einem einzelnen Thread auf einem Intel®Core™2 Duo P8600 Prozessor durchgeführt.

Erwartungsgemäß ist der adaptive slicing Prozess deutlich schneller als der statische, da die Anzahl der Schichten deutlich reduziert wird. Ist die Anzahl der Schichten gleich, braucht die Analyse des Modells signifikant zusätzliche Zeit. Diese ist aber im Sekunden- bis Minutenbereich und im Verhältnis zur Druckzeit nur geringfügig. Die Druckzeit bleibt somit der dominierende Faktor.
Der slicing Prozess, also das Generieren der Schichten, dauert bei adaptivem Slicing verhältnismäßig deutlich länger. Die Laufzeit der Slicing-Routine im 3. Modell erklärt aber mit $\sim$7% bei Weitem nicht die gesamte Zunahme der Zeit. Dies kann an den Eigenschaften des Modells liegen, etwa wenn durch die Konzentration der Schichten in komplexen Bereichen das Berechnen der Extrusionspfade rechenintensiver wird.

Modell	Konfiguration	Laufzeit [s]	Slicing [%]	Cusp [%]	Extrusions-breite [%]
L-Profil	Statisch [0.1 mm]	2.03	3.62	-	-
	Statisch [0.253 mm]	**0.94**	2.96	-	-
	Adaptive Schichtdicke	0.89	7,53	2.5	-
	Adaptive Extrusion	0.92	6.54	-	3.36
	Voll adaptiv	**0.97**	12.25	3.06	3.83
Servo-Modul	Statisch [0.1 mm]	38.98	0.79	-	-
	Statisch [0.172 mm]	**23.67**	0.81	-	-
	Adaptive Schichtdicke	22.3	7,11	5.01	-
	Adaptive Extrusion	25.81	4.19	-	3.29
	Voll adaptiv	**24.31**	10.19	4.89	3.01
X-Carriage	Statisch [0.1 mm]	321.00	0.40	-	-
	Statisch [0.258 mm]	**114.31**	0.38	-	-
	Adaptive Schichtdicke	138.32	2.87	1.92	-
	Adaptive Extrusion	144.35	4.31	-	3.77
	Voll adaptiv	**193.84**	7.33	2.00	3.82

Tabelle 6.5. – Vergleich der Laufzeiten bei statischem und adaptivem Slicing für 3 Modelle unterschiedlicher Komplexität. Sowohl die Berechnung der Schichthöhe als auch die der Extrusionsbreite findet während des Slicing-Prozesses statt, der die einzelnen Layer generiert und füllt. Die Spalten „Cusp“ und „Extrusionsbreite“ sind Komponenten von „Slicing“.

Die Struktur von **Slic3r** wird zum Zeitpunkt der Verfassung dieser Arbeit überarbeitet und in weiten Teilen konsequenter nach C++ portiert. Es ist zu erwarten, dass sich das Laufzeitverhalten in absehbarer Zeit deutlich verändern wird.

6.2.2. Parallelisierbarkeit

Da die Verarbeitung komplexer Modelle rechenintensiv ist, spielt Parallelisierung eine wesentliche Rolle um die Verarbeitungszeit niedrig zu halten. **Slic3r** unterstützt die Verwendung von Threads und kann viele Teilaufgen parallel verarbeiten, wenn genügend Speicher vorhanden ist. Bei adaptivem Slicing kann die Höhe der nächsten Schicht immer erst berechnet werden wenn die Höhe der vorherigen bekannt ist, daher ist Parallelisierung nicht unmittelbar möglich. Prinzipiell kann die Analyse der Oberflächen in einem seriellen Schritt erfolgen und die eigentlichen Schnittoperationen anschließend parallel ausgeführt werden. Dies ist aber nicht möglich wenn durch die Bestimmung der Extrusionsbreite der Extruder gewechselt werden kann, da die Schichtdicke bei kleineren Extrudern reduziert werden muss. Die Bestimmung der Schichtdicke und Extrusionsbreite muss daher alternierend erfolgen.

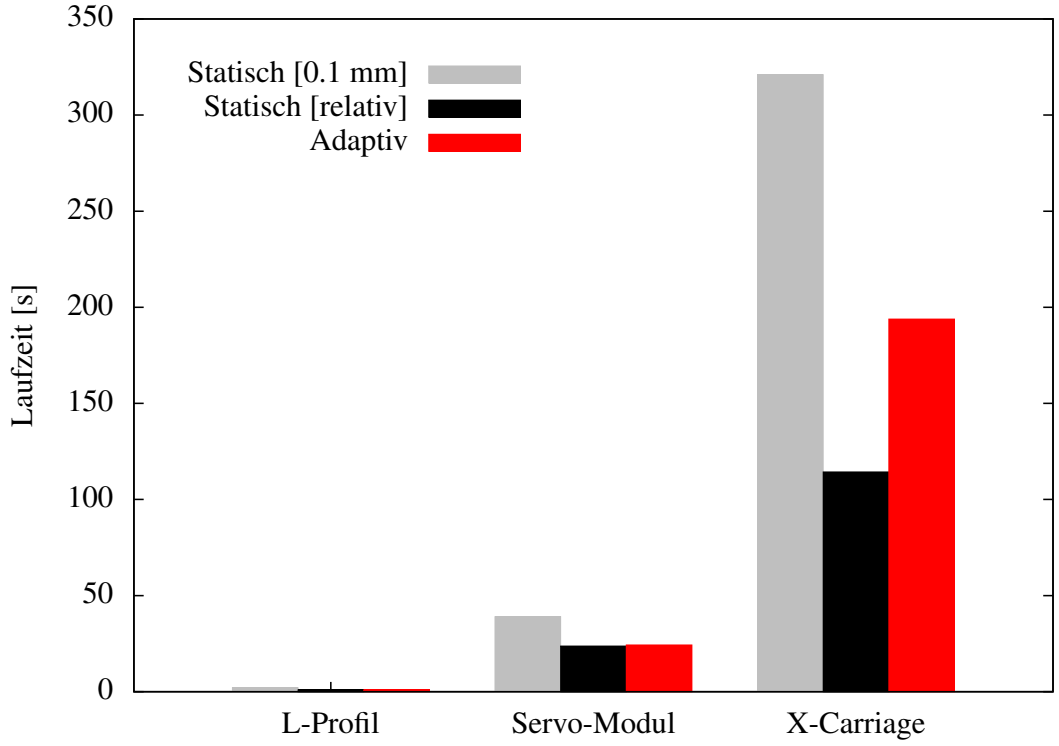


Abbildung 6.7. – Absolute Laufzeit bei adaptivem und statischem Slicing.

Tyberg [Tyb98] verwendet dickere Slabs mit festgelegter Ober- und Untergrenze, innerhalb derer die Schichtdicke variiert. Diese Slabs sind parallelisierbar, schränken aber die Effizienz der adaptiven Berechnung ein, da der Algorithmus an den Slab-Grenzen vordefinierte Ebenen treffen muss. Die Dicke der Slabs entspricht bei Tyberg dem Durchmesser des Extruders, die Anzahl festgelegter Schichtgrenzen ist daher unverhältnismäßig hoch.

Um den Effekt der bereits vorhandenen Parallelisierung abschätzen zu können, wurde das `X-Carriage` Objekt ein weiteres mal statisch und adaptiv verarbeitet, dieses mal in 2 Threads:

Konfiguration	Laufzeit [s] 2 Threads	Laufzeit [s] 1 Thread	Slicing [%]	Cusp [%]	Extrusions-breite [%]
Statisch [0.258 mm]	**107.90**	114.31	0.4	-	-
Adaptiv	**152.33**	193.84	7.36	2.00	3.82

Tabelle 6.6. – Laufzeit des X-Carriage Modells bei paralleler Verarbeitung.

Die Laufzeit reduziert sich, der auf adaptives Slicing entfallende Anteil bleibt aber im Rahmen der Messgenauigkeit konstant. Die weitere Analyse ergibt, dass der dominierende Teil der Laufzeit auf das Generieren des *G-Codes* aus den auf Layer-Ebene aufbereiteten Polygonen entfällt.

Dieser Teil ist aber bislang nicht parallelisiert. Für die Gesamtlaufzeit spielt adaptives Slicing daher bislang nur eine untergeordnete Rolle, da die Effizienz anderer Codebestandteile dominiert. Werden mehrere Objekte gleichzeitig verarbeitet, kann trivialerweise pro Objekt parallel gerechnet werden.

7. Fazit

In dieser Arbeit wurden existierende Techniken für adaptives Slicing weiterentwickelt und implementiert. Damit wurde sowohl die erreichbare Qualität als auch die Druckzeit signifikant verbessert. Alle erarbeiteten und umgesetzten Methoden wurden innerhalb der bestehenden Open Source Software **Slic3r** umgesetzt und sind damit offen verfügbar, direkt einsetzbar und können als Grundlage für weitere Verbesserungen dienen. Der Effekt der Veränderungen wurde sowohl theoretisch abgeschätzt als auch empirisch verifiziert.

7.1. Was wurde erreicht

Es wurde eine adaptive Berechnung der Schichtdicke auf Basis des Cusp-Fehlermaßes implementiert. Diese wurde so erweitert dass sie vorausschauend Änderungen der Oberflächenneigung und horizontale Oberflächen erkennt und die Dicke einer Schicht global korrekt berechnet. Dazu wird zusätzlich zur Oberflächennormalen eines Facets auch die vertikale Distanz stärker geneigter Facets mit in die Berechnung einbezogen.
Um zusätzlich die Qualität der erzeugten Oberfläche, vor allem an vertikalen Flächen zu berücksichtigen, wurde eine Analyse der Oberflächenstruktur gedruckter Objekte durchgeführt. Es wurde ein linearer Zusammenhang zwischen Schichtdicke und Oberflächenfehler identifiziert, dieser ist unabhängig vom Durchmesser des Extruders. Die sich hieraus ergebene, triviale Berechnung der Schichtdicke auf Basis der erwünschten Oberflächengüte, wurde mit der Cusp-basierten Berechnung zusammengeführt.

Ein ein auf der Definition von r-regulären Umrissen basierender Ansatz wurde entwickelt, der die Berechnung einer optimalen Extrusionsbreite für ein Polygon erlaubt. Dies ermöglicht es, eine adaptive horizontale Auflösung zu verwenden um hohe Druckgeschwindigkeiten auch bei filigranen Objekten zu ermöglichen. Es wurde gezeigt dass dieser Ansatz auf zwei oder mehr Extruder anwendbar ist, um das Intervall der möglichen Extrusionsbreiten deutlich zu vergrößern.

7.2. Ausblick

Mit dieser Arbeit wurde die Basis für eine offene Implementation adaptiver Slicing Algorithmen gelegt. Es wurden zentrale bekannte Techniken auf den aktuellen Stand der Entwicklungen angewendet, zusammengeführt und durch eigene Ansätze erweitert. Damit diese Techniken ihre volle Wirkung entfalten können, ist es sehr wünschenswert, sie auf weitere Aspekte des Slicing-Prozesses anzuwenden. Zu nennen sind hier insbesondere:

Lokal adaptive Extrusionsbreite Bislang wird die Berechnung der Extrusionsbreite einheitlich für alle Polygone einer Schicht durchgeführt. Insbesondere bei verzweigenden Objekten oder solchen, die Löcher enthalten, wäre es optimaler, für jedes Polygon eine Extrusionsbreite zu generieren. Zwischen den einzelnen Perimetern existieren Abhängigkeiten, da zwischen ihnen Grenzen festgelegt werden müssen, an denen sie sich berühren. Diese Abhängigkeiten machen den Suchraum komplex und erfordern für eine effiziente Lösung den Einsatz von Vereinfachungen und Heuristiken. Zudem müssen die `extruder`- und `flow`-Objekte auf Polygonebene lokalisiert werden.

Adaptives Füllen Eine adaptive Schichthöhe wird bislang nur für den Außenbereich des Objekts verwendet. Im Innenbereich ist die Schichtdicke limitiert auf Vielfache der Perimeterschichten. Dieser Ansatz ist besonders bei dickeren Perimeterschichten ineffizient. Ist die Schichtdicke beispielsweise etwas größer als die Hälfte des Extruderdurchmessers, kann im Innenbereich überhaupt nicht kombiniert werden.

Vereinfachte Konfiguration In **Slic3r** existieren momentan deutlich über 100 Optionen, die den Slicing-Prozess beeinflussen. Der Lernaufwand für Benutzer ist beträchtlich und es ist für nahezu jeden Druckvorgang eine manuelle Anpassung einiger Parameter nötig. Adaptive Algorithmen kennen die geometrische Struktur eines Objekts und bieten somit das Potential, zentrale Parameter auf Basis des aktuellen Objekts automatisiert zu setzen. Dieser Schritt ist essentiell auf dem Weg zu einer automatischen Verarbeitung, wie wir sie etwa von Papierdruckern gewohnt sind.

Abkürzungsverzeichnis

CSG Constructive Solid Geometry
FDM Fused Deposition Modelling
LEC Largest Empty Circle
MJM Multi Jet Modelling
PLA Polylactic Acid - Polymilchsäure
PWID Pairwise Interference Detection
SLA Stereolithography
SLM Selective Laser Melting
SLS Selective Laser Sintering
STL Surface Tesselation Language oder Standard Triangulation Language

A. Anhang

A.1. Volumetrischer Fehler

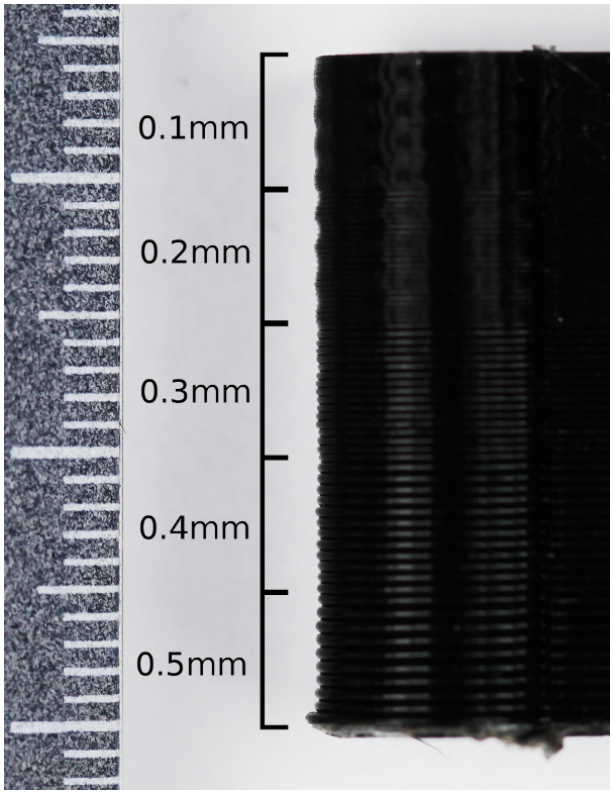


Abbildung A.1. – Regionen unterschiedlicher Schichtdicke im Testobjekt. Die wellenförmige Oberfläche im oberen Bildbereich ist eine Folge der unsauberen Rotation der Z-Achse.

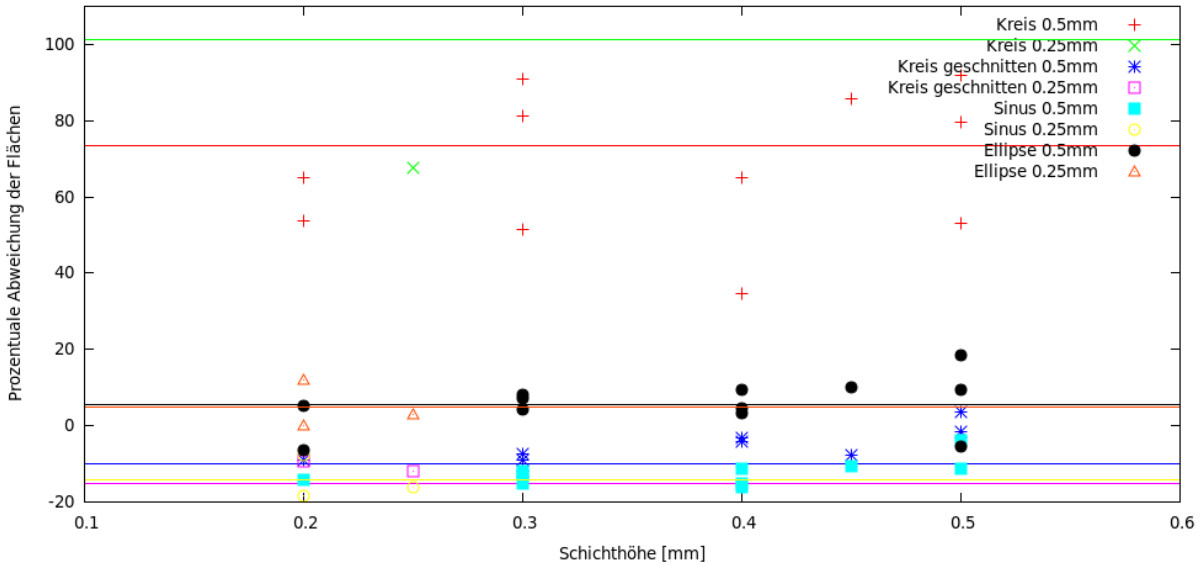


Abbildung A.2. – Prozentuale Abweichung der Integrale zwischen gemessener Oberfläche und geometrischem Primitiv. Die Abweichungen wurden pro Objekt für jeweils einen Bereich gleicher Schichtdicke gemittelt.

A.2. Extrusionsfaktor

Zur Bestimmung der Grenzwerte für den Extrusionsfaktor f_e wurden zwei Testreihen durchgeführt. Eine mit einem geschlossenen Block zum Erzeugen einer geschlossenen Oberfläche (Abb. A.3), eine mit einem Objekt mit einer so geringen Wandstärke, dass nur ein einziger Perimeter gedruckt wurde zum Erzeugen schmaler Wände (Abb. A.4). Bei einem Multiplikator von 1.5, sowohl für die Untergrenze ($\frac{\varnothing}{1.5}$) als auch für die Obergrenze ($\varnothing \times 1.5$), wurden für beide Objekte mit allen Filamenttypen ausreichende Ergebnisse erzielt. Ist f_e zu groß, kommt es zu sehr rauen Oberflächen bzw. abreißendem Filamentfluss (Abb. A.4 rechts). Die Verarbeitungseigenschaften der Filamenttypen unterschiedlicher Hersteller können stark voneinander abweichen.

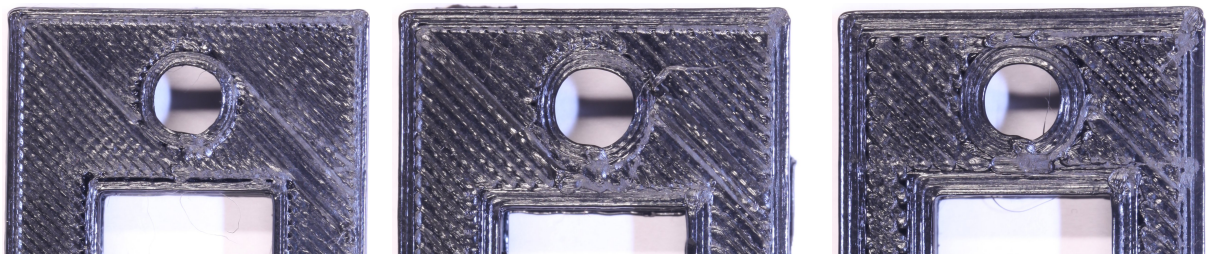


Abbildung A.3. – Effekt verschiedener Extrusionsfaktoren. Extrusionsbreite von links nach rechts: 0.33mm, 0.5mm, 0.75mm. Alle Objekte wurden mit dem selben 0.5mm Extruder gedruckt.

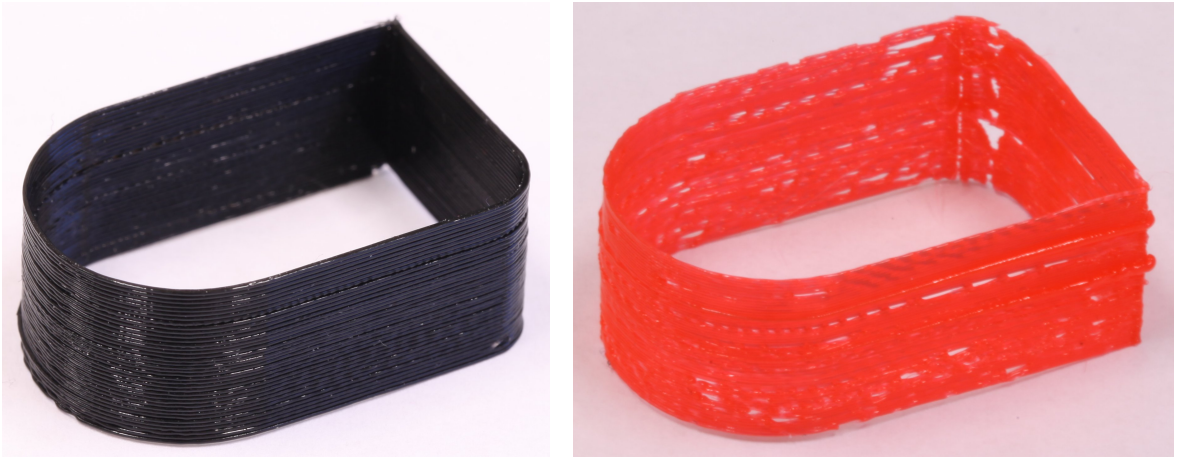


Abbildung A.4. – Erzeugung dünner Wände mit einem geeigneten Extrusionsfaktor (f_e = 1.5) links, und einem zu großen (f_e = 2.0), bei gleichzeitig schlechterer Filamentqualität rechts.

Literaturverzeichnis

[BA92] BRANDT, J. W. und V. R. ALGAZI: *Continuous skeleton computation by Voronoi diagram*. CVGIP: Image Understanding, 55(3):329 – 338, 1992. 36

[BRA+] BROOKS, H., A. E. W. RENNIE, T. N. ABRAM, J. MCGOVERN und F. CARON: *Variable fused deposition modelling - concept design and tool path generation*. In: *12th Rapid Design, Prototyping & Manufacturing Conference*. 22, 36, 38, 57

[Bur89] BURNS, M.: *StereoLithography Interface Specification*. 3D Systems, Inc., 1989. 16

[CLL03] CHUA, C. K., K. F. LEONG und C. S. LIM: *Rapid Prototyping: Principles and Applications*. World Scientific Publishing Co. Pte. Ltd., 2003. 16

[dB10] BRUJIN, E. DE: *On the viability of the Open Source Developement model for the design of physical objects*. 2010. 11

[DM94] DOLENC, A. und I. MÄKELÄ: *Slicing procedures for layered manufacturing techniques*. Computer-Aided Design, 26(2), 1994. 19, 20, 30

[FA94] FAROUKI, R.T. und S.R. ABRAMS: *Offset curves in layered manufacturing*. Manufacturing science and engineering, 68(2), 1994. 21

[Hub12] HUBER, S.: *Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice*. Shaker Verlag, Aachen, 2012. 38

[J. 12] J. MOILANEN AND T. VADÉN: *Manufacturing in motion: first survey on 3D printing community* . `http://surveys.peerproduction.net/2012/05/manufacturing-in-motion`, 2012. [Stand 15.02.2013]. 11

[JHS+11] JONES, R., P. HAUFE, E. SELLS, P. IRAVANI, V. OLLIVER, C. PALMER und A. BOWYER: *RepRap - the replicating rapid prototyper*. Robotica, 29:177–191, 2011. 11, 24

[KWHZ14] KRUPKE, D., F. WASSERFALL, N. HENDRICH und J. ZHANG: *Printable Modular Robot: An Application of Rapid Prototyping for Flexible Robot Design*. 2014. 60, 63

[PC03] PARK, S. C. und Y. C. CHUNG: *Mitered offset for pro®le machining*. Computer-Aided Design, 35(5):501 – 505, 2003. 38

[PRD03a] PANDEY, P. M., N. V. REDDY und S. G. DHANDE: *Improvement of surface finish by staircase machining in fused deposition modeling*. Journal of Materials Processing Technology, 132, 2003. 21

[PRD03b] PANDEY, P. M., N. V. REDDY und S. G. DHANDE: *Real time adaptive slicing for fused deposition modelling*. International Journal of Machine Tools & Manufacture, 43, 2003. 22, 51

[PRD03c] PANDEY, P. M., N. V. REDDY und S. G. DHANDE: *Slicing procedures in layered manufacturing: a review*. Rapid Prototyping Journal, 9, 2003. 14, 19

[RW91] ROCK, S. J. und M. J. WOZNY: *Utilizing Topological Information to Increase Scan Vector Generation Efficiency*. 1991. 17

[Sab96] SABOURIN, E.: *Adaptive high-precision exterior, high-speed interior, layered manufacturing*. 1996. 21, 29

[Ser82] SERRA, J.: *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982. 36

[TFBA98] TATA, K., G. FADEL, A. BAGCHI und N. AZIZ: *Efficient slicing for layered manufacturing*. Rapid Prototyping Journal, 4, 1998. 19, 46

[Tyb98] TYBERG, J. T.: *Local adaptive slicing for layered manufacturing*. 1998. 17, 20, 21, 29, 65

Verzeichnis der Webadressen

[1] E3D Repository. https://github.com/revk/E3D, Sep 2011. [Stand 27.01.2014].

[2] Prusa Mendel I Dokumentation. http://reprap.org/wiki/Prusa_Mendel_(iteration_1), Sep 2012. [Stand 23.01.2014]. 24

[3] Skeinforge Projektwebsite. http://fabmetheus.crsndoo.com/, März 2012. [Stand 27.01.2014]. 27

[4] Kisslicer Herstellerwebsite. http://www.kisslicer.com, Mai 2013. [Stand 27.01.2014]. 28

[5] RepRapPro Slicer Dokumentation. http://reprap.org/wiki/RepRapPro_Slicer, Aug 2013. [Stand 27.01.2014].

[6] Ultimaker B.V. CuraEngine Repository. https://github.com/Ultimaker/CuraEngine. [Stand 28.01.2014]. 27

[7] Netfabb GmbH. Netfabb Herstellerwebsite. http://www.netfabb.com, 2014. [Stand 27.01.2014]. 28

[8] Angus Johnson. Clipper Polygon Bibliothek. http://www.angusj.com/delphi/clipper.php, Jan 2014. [Stand 28.01.2014]. 27, 39

[9] LLC MakerBot Industries. Makerware Herstellerwebsite. http://www.makerbot.com/makerware/, 2014. [Stand 27.01.2014]. 28

[10] Alessandro Ranellucci. Slic3r Projektwebsite. http://slic3r.org. [Stand 28.01.2014]. 25, 42

Selbstständigkeitserklärung

Ich versichere, dass ich die Masterarbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ich bin damit einverstanden, dass die Arbeit in den Bestand der Bibliothek des Departments Informatik aufgenommen wird.

Ort, Datum Hamburg, 28.02.2014

Unterschrift: _____

Florens Wasserfall