TU WIEN Informatics

3D-Drucken zum Verständnis Komplexer Topologien

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Florian Staudacher
Matrikelnummer 0825724

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dr.techn. Alexey Karimov

Wien, 18. März 2022

Florian Staudacher	Eduard Gröller

TU WIEN Informatics

3D-Printing to Understand Complex Topologies

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Florian Staudacher
Registration Number 0825724

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dr.techn. Alexey Karimov

Vienna, 18th March, 2022

Florian Staudacher	Eduard Gröller

Erklärung zur Verfassung der Arbeit

Florian Staudacher

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18. März 2022

Florian Staudacher

Kurzfassung

In Bildung und Lehre kann es oftmals ausschlaggebend sein, wie ein Thema oder Konzept präsentiert wird, um den Lernerfolg von Schülern konstruktiver und einfacher zu gestalten. Das Ziel dieser Arbeit ist es, einen komplexen Gegenstand greifbar zu machen. Speziell geht es um das dreidimensionale Erscheinungsbild eines mathematischen Ausdruckes, der eine gewisse Topologie beschreibt und wie diese von ihren Parametern beeinflusst wird.

Der gewählte Zugang umfasst *3D-Fertigung* solcher Gegenstände, als Versuch, ein hilfreiches Werkzeug anzubieten, um die Besonderheiten dieser Objekte verständlich zu machen. Zu diesem Zweck wurde im Rahmen dieser Arbeit eine Softwareanwendung erstellt, die es ermöglicht, anhand einer Vorschau des Objektes die Parameter zu manipulieren. Sobald die Gestalt des Objektes den Vorstellungen des Anwenders entspricht, können die Daten in einem passenden Format exportiert werden, um weiter verarbeitet und schließlich "3D-gedruckt" zu werden.

Die Software zur Eruierung der Machbarkeit und auch alle weiteren Werkzeuge, welche im Rahmen dieses Dokuments erwähnt werden, können dazu genutzt werden, um physische 3D-Gegenstände aus den beschriebenen Modellen zu erzeugen. Diese Instrumente können als Ausgangspunkt gesehen werden, um das Thema weiter auszuführen, dennoch sind auch Herausforderungen deutlich geworden, die sich von der Auswahl der Objekte, der Vorbereitung zur Fertigung bis hin zu den realen Produkten gezogen haben. Aufbauende Arbeiten können diese Schwierigkeiten berücksichtigen und gegebenenfalls bessere Ansätze zur Lösung finden sowie sich von den vorgeschlagenen Ideen inspirieren lassen.

Abstract

In education and teaching, the way a topic or a concept is presented can be an important factor in how easily and thoroughly it can be grasped by students. The goal of this thesis project is to make a complex subject tangible, in this case the three-dimensional appearance of a mathematical term describing a certain topology and the way it is shaped by its parameters.

The approach involves *3D-fabrication* of such artefacts in an attempt at providing a helpful tool to comprehend the peculiarities of such objects. A software program was created to preview and manipulate topological objects from their parameterized representations as part of this thesis. Once the shape resembles the intentions of the user, the data can be exported in a format that is suitable for further processing and finally "3D-printing".

The proof-of-concept software as well as other auxiliary tools described in this document can be used to produce physical 3D-artefacts of the described models. While it might be seen as a starting point to elaborate further on this topic, some issues become apparent throughout each of the steps leading from the selection of the object, the preparation before fabrication all the way to the real-world artefacts that were created. Any future work might want to consider these issues and possibly find better solutions to work around given limitations, and maybe find inspiration in the suggested propositions.

Contents

CHAPTER 1

Complex Topological Objects

All throughout primary and secondary education stages, even for job specific training or while studying for an occupational career in science and research, people will get in contact with mathematics and other closely related disciplines. A significant part of the required education attempts to convey to the people pursuing those kinds of careers a deeper understanding of *objects*, how they are *shaped*, *organized*, and how those shapes can be *described in mathematical terms*.

This kind of representation is required to relate the objects dimensions and to make accurate predictions about how little changes in parameters will cause non-obvious changes in the mathematical entities. One of the simple examples to demonstrate the task of relating the area of a rectangle to the length of either of its edges. Such an example might not have much necessity in the real world, but most things we handle on a day-to-day basis can be described geometrically. In the field of education specifically, the tasks might manifest in calculation of areas or circumferences in two dimensions, or students might be required to intersect planes and objects in three-dimensional space and calculate volumina of complex shapes represented by mathematical terms. Especially when students later seek occupations in the field of engineering, the abstract volume of an irregular shape can actually represent something with a lot of impact like the capacity of a river bed or a dam.

Beware that research and most of the work for this thesis was done around 2017 and thus might reflect an already obsolete state of the art. 3D-fabrication and the area of expertise as well as the software tools used in the process are rapidly evolving and therefore some of the items discussed in this document might be already outdated.

1.1 Motivation

The shape of an object encoded in such an abstract form is *difficult to grasp* just by looking at the equation form. This might make it harder to relate to the matter at hand and impede the process of comprehending an abstract object, especially when students find it problematic to picture a given object simply by parsing the mathematical representation or by looking at a graphical cross section in 2D. This circumstance might be amplified the more complex objects will get during the course of the academic career, or when students have not formed a mental image of the figure they are supposed to handle in their studies. Prior to an actual three-dimensional representation there exists only a mathematical term, for which the goal is to be explored, and to find any previously unknown intricacies.

To facilitate the learning process and help in understanding complex shapes it can be advantageous that the object at hand can be viewed visually, either in a 2D representation (as a 2D projection of a 3D-object) on paper or on a computer screen. While paper classically has been the medium of choice for teaching materials, computers increasingly find their way into classrooms. Their use as a tool that aids in teaching opens up *new ways of interaction with the learning matter*. Computers can not only be utilized for presenting slides with a projector, instead, students can gain access to carefully prepared digital learning materials, which can be made interactive in order to offer an added incentive for working with them. In the context of objects with complex topology and shapes this gives the user the possibility to interact with the viewed object via rotating, zooming, and panning metaphors or even to modify the object in ways that would not be possible otherwise like slicing it along arbitrary axes or viewing it as exploded representation.[KLM+10]

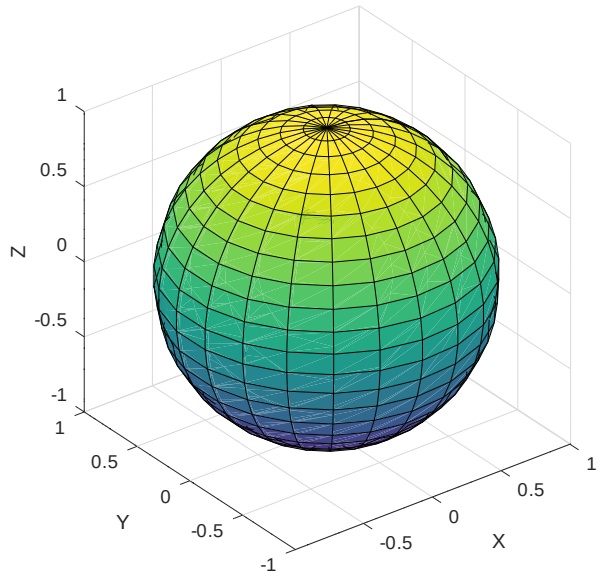
To improve comprehension of purely abstract mathematical descriptions, we suggest to translate a visual representation directly into the physical world via the accompanying software to this thesis. By computing abstract objects in a certain parameter space we create a 3D-description of abstract terms describing 3D-objects. The various points along the object's surface are calculated in XYZ-space, forming a virtual 3D-model in computer memory. The computer model can then be rendered to the screen by the software, while still allowing the input parameters of the terms to be modified.

In a subsequent step, the surface description is exported in a suitable file format followed by post-processing in the control application of a 3D-fabrication device, also commonly known as a *3D-slicing software*. The software performs a few optimization steps, then iterates over the 3D-object layer by layer and intersects it with horizontal planes to produce the information for the fabricator, often referred to as **3D-Printer**. This information is then translated to actual machine and motor instructions, including the temperature of the material extruder and the exact degrees the motors have to rotate in order to move along the axes while depositing the fabrication material. Some common file formats for these intermediate steps will be presented briefly in a later chapter of this thesis.

Finally, the goal of this process is to create an actual, tangible object that can be examined by *interacting with it physically and naturally*, immediately allowing the users to not only form a mental image of a shape, but also to allow them to *reveal all the intricacies* of more complex forms, they would not have been able to notice at a glance, otherwise.

1.2 Object Surface Definition

Models suitable for 3D-fabrication are often represented by a parameterized equation that yields 3D-coordinates for every sampled position in the parameter space (see Figure 1.1 for an example term of a parameterized sphere). Such a form can, in turn, be used to calculate the vertices of the object in $\mathbb{R}^3$ using the correct lower and upper bounds for ϕ and θ. In case of the sphere the steps for θ are bounded between 0 and π, and ϕ between 0 and 2π.



(a) Sphere plotted using the parameter space: $\theta = [0, \pi]$ and $\phi = [0, 2\pi]$.

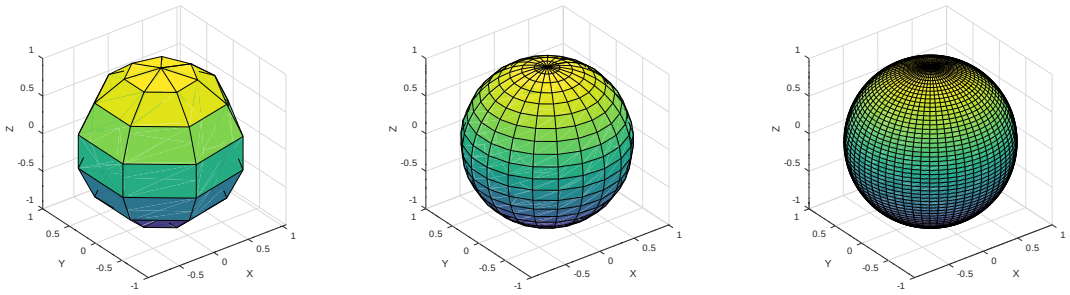
$$x = r \cdot \sin(\theta) \cdot \cos(\phi)$$
$$y = r \cdot \sin(\theta) \cdot \sin(\phi)$$
$$z = r \cdot \cos(\theta)$$

(b) Parameterized term for a sphere.

Figure 1.1: Sphere in mathematical and graphical representations.

The number of sampling positions inside the parameter space influences the quality of the resulting 3D-model, since those values are the basis for interpolating the object surface (see Figure 1.2). For practical reasons the software created during this thesis limits the number of vertices produced by sampling of the surface to a maximum of a few hundred thousand, in order to ensure a reasonable rendering time and export file size. In theory

there would be no limit on how small the intermediate steps could be placed apart from each other, though, it would make no sense producing a 3D-model that has a far higher resolution than the fabrication process could produce. On the other hand, lowering the resulting vertex count to under about one hundred also yields bad results, since the object cannot be depicted accurately anymore, depending on what form is currently selected.



(a) Sphere plotted using 8 interpolation steps for ϕ and θ.

(b) Same as Fig. 1.2a, but with 22 steps.

(c) Same as Fig. 1.2a, but with 60 steps.

Figure 1.2: Sphere in various densities of interpolation, showcasing the increase in quality by using more interpolation steps. An increase in steps beyond what is portrayed in Fig. 1.2c would be almost negligible to the naked eye.

Furthermore, some refinement during the sampling process is needed, because simply looping over the surface of an object using ϕ and θ would produce *trapezoids* ("quadrilaterals"). Most computerized 3D-modeling tools, the file format used for exporting the model, and also 3D-Graphics APIs like OpenGL - a software layer used for rendering the preview inside the application - are optimized for handling 3D-models defined by **triangles**, not trapezoids. This makes it necessary for the sampling algorithm to create (at least) two triangles instead of one quadrilateral cell of the parameter space grid. For the resulting model this means that the number of faces is doubled. Therefore C++, a compiled programming language, is used for the calculation of the model surface and the preview is rendered using OpenGL with hardware acceleration, which enables a quick reaction time to altered parameters and a very fine grained approximation based on tens to hundreds of thousands of vertices.

Not every object is suitable for getting turned into a 3D-fabricated part by the automated process of generating a surface representation used in the software. There are a few requirements that have to be met in order for a parametric model to represent a closed volume.

Most importantly the surface of the object has to represent a solid object with a real thickness that is greater than the smallest resolution unit of the fabrication device, so it does not become a volume-less flat shape. Otherwise the respective part would vanish during the slicing process and the object might not be produced correctly, or, in the worst case scenario, the structure would be broken and the additive layering would fail.

The same problem arises for intricate surface details, which fall below the possible feature size of the device. This has to be handled in respect to the configuration of the individual manufacturing device, since these features might vary in size, but a common diameter for the tip of the deposition head is ≈ 0.4 mm, which can be moved to a precision of around 600 Dots per Inch (DPI). Surface quality is also influenced by the layer height - any curvature along the z-axis is only an approximation and will produce a "stair-stepping" effect. A common height that can be used while still producing models at a reasonable time is ≈ 0.2 mm to ≈ 0.1 mm [GZR+15], which can be further improved afterwards by sanding or chemically dissolving the outermost layer of the model and letting it re-solidify, to achieve a smooth finish.

The main requirement for the model is to be a *manifold surface* - a surface enclosing a volume. This is illustrated by the "Sombrero" model (see Figure 4.4), which per se describes a surface without a volume. In order to meet the requirement, some post-processing steps need to be added to transform the model into an appropriate object. This will be explained in greater detail in Section 4.2.

Also, the requirement of the model being a *closed volume*, also called a **watertight** mesh surface, can be easily clarified by using the metaphor it is named after: if someone was to fill the model with water, would any of it leak out? In mathematical terms, this means that every edge of the mesh has to be adjacent to two faces, which ensures the object encloses a volume and has no holes.

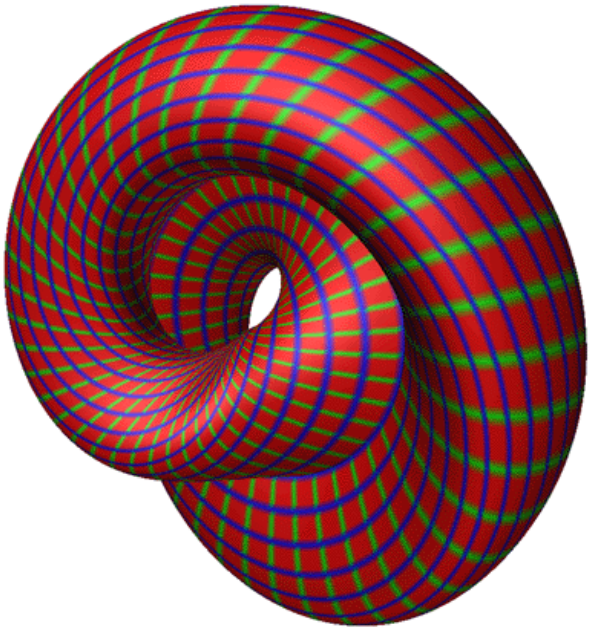
This also calls for all the *surface normal vectors* having to point outwards in order to coherently define the *inside* of the model. Some of the more sophisticated slicing programs attempt to fix all such defects in the mesh definition automatically upon importing the model, but there are still cases, when such repair can fail. For example, the **Klein Bottle**, included into the software application of this thesis, cannot be fixed by such repairing techniques, since its topology is based on the **Möbius strip**. This causes the *inside* of the object to turn into the *outside* at some point, making any efforts of aligning normal vectors futile. That specific model only becomes printable once a section is removed and it is no longer a continuous strip.

Another object that bears properties which require some relaxation prior to the 3D-printing in regard to the normal vectors is the *Figure-8 immersion*, pictured and specified by the equation presented in Figure 1.3. Originally the mesh is self intersecting, since the Figure-8 immersion is rotated around its perimeter, and half of the model is turned inside out. During model creation in the application, half of all normal vectors are inverted to point into the opposite direction. This simplification allows the slicing program to attempt its repair algorithms without problems.

An aspect to be concerned about while printing complex topologies is that the object should present all noteworthy features on the outside, since all enclosed features will not be visible, unless the model is sliced open along a plane during post-processing or an exploded view is produced. On the other hand, if during fabrication a transparent material is used, possibly in conjunction with a second, colored polymer, some effects

$$x = (r + \cos(n \cdot \frac{\theta}{2}) \cdot \sin(\phi) - \sin(n \cdot \frac{\theta}{2}) \cdot \sin(2\phi)) \cdot \cos(m \cdot \frac{\theta}{2})$$
$$y = \sin(n \cdot \frac{\theta}{2}) \cdot \sin(\phi) + \cos(n \cdot \frac{\theta}{2}) \cdot \sin(2\phi)$$
$$z = (r + \cos(n \cdot \frac{\theta}{2}) \cdot \sin(\phi) - \sin(n \cdot \frac{\theta}{2}) \cdot \sin(2\phi)) \cdot \cos(m \cdot \frac{\theta}{2})$$

(a) Parameterized term for the Figure-8 immersion. r is the size of the radius; m specifies the number of revolutions around the perimeter and n defines how many twists there will be around the center axis.



(b) Rendering of the Figure-8 immersion highlighting the parameters θ (green) and ϕ (blue) as isolines.

Figure 1.3: Mathematical formulas used to describe the *Figure-8 immersion* used throughout this thesis and the accompanying illustration highlighting the grid of spherical coordinates using isolines. Source: [Bou96]

might be achievable, highlighting the enclosed features by viewing them through the transparent outer shell. This idea is explored further in Section 4.3.

CHAPTER 2

3D-Artefact Fabrication

In geometry, there are many different kinds of objects that can have a parameterized mathematical description. A one-dimensional manifold (this includes, for example, lines and circles) is theoretically suitable for fabrication, but producing two-dimensional objects with a device capable of generating 3D-objects might arguably defeat its purpose. The kind of object that can be created using the currently available consumer-level fabrication processes mainly consist of *two-dimensional manifolds*, also called **surfaces**. The geometric description of these surfaces have to abide by some rules, so the objects can be successfully processed by a software program. Some of these rules were already discussed in the previous chapter. Those restrictions (solid object, minimum feature dimension, closed volume, well-defined surface normal vectors) are necessary in order to allow conversion of the model description into the fabrication instructions for the fabrication device, which, in the end, will produce a tangible, physical artefact. Also, other kinds of objects, including those defined in higher-dimensional spaces, are essentially unsuitable for the fabrication method discussed, unless they are projected into 3D-space and properly sampled.

The most important requirement for the object to be printed is its *orientability*. This feature is necessary for the fabrication device software to correctly determine, what parts of the object should be considered inside and thus belong to the volume that is to be filled with material. If the surface does not fulfill this requirement, there are generally some ways to still produce a geometric model suitable for 3D-Artefact fabrication. One way to achieve this is to enclose the object in a mesh that follows the surface on the outside, producing a *convex hull*, similar as throwing a blanket over a sculpture. This will produce a geometry capable of being handled by a 3D-fabricator, but during this process all internal features of the object are lost. Also this adds an additional intermediate processing step, which was not included in the software for this thesis.

2.1 Physical Fabrication Process

A variety of objects can be produced utilizing the general fabrication technique of additive layering. Depending on the material that is used in the process and also the mode of operation of the fabrication device, the deposition of layers is generally done in two ways.

Either this is achieved by depositing a thin level of substrate over the entire build surface, which is then bound in the desired location. The remaining material can be removed and used again, subsequently. The other way of layering can be considered to be more precise since the material is only deposited where it needs to be for creating the final object. This second approach uses the printing material more sparingly, but also opens up a few challenges in how the fabricated object needs to be prepared. For example, it is necessary to add support structures underneath overhanging areas of the object, since the depositing process would otherwise occur in mid-air with no material to layer upon, and the result would be wrong, warped or otherwise undesireable.

In recent time, 3D-fabrication of tangible artefacts (also, often referred to as "3D-printing") has gained much interest from commercial manufacturing businesses and private individuals alike. The process is used for rapid prototyping of Computer Aided Design (CAD) models and produces production-ready parts that could not otherwise be fabricated by other methods, e.g., for industrial purposes as well as comparatively mundane items such as keyring pendants and control knob handles by enthusiastic hobbyists. Although, especially for engineering purposes, where more conventional fabrication methods would require setting up production lines with lots of expensive equipment or time- and labor-intensive specialized production procedures, this approach of producing parts straight from the CAD software can save a lot of time and money.

The range of materials used for the 3D-printing varies depending on the versatility of the designated use-case. The most common material in the low-end and consumer-level spectrum is certainly plastic. There are many different chemical compounds that fall under that category, some of which may sound familiar: Polylactic Acid (PLA), Acrylonitrile butadiene styrene (ABS), Polyethylene terephthalate (PET), or Nylon - just to name a few. For 3D-fabrication, plastic is typically heated to around 200°C and the molten material is then funnelled through an extruder to create an object layer by layer. The exact specifications may vary by fabricator models, but a layer height of 0.1 mm and a depositing speed of about 40-50 mm/s is well within range of most 3D-printers in the lower price range. This is also the environment, where most of the commercially available equipment for crafting hobbyists and technology enthusiasts is available.

The aforementioned chemical compounds can be blended with other materials like wood or carbon fibers, which have a positive effect on the structural integrity of the printed object and gives it specific desired surface properties. Different characteristics can be achieved by the use of plastic, like adding fluorescent pigment to the material or making the compound transparent to achieve a luminous or see-through effect.

A different approach using a photosensitive fluid resin curable by ultraviolet light, as a

consequence, requires the fabrication to proceed from the top down instead of from the bottom up. This process is also called Stereolithography (SLA) and was first patented in 1986 [Hul86] and later commercialized by the company 3D Systems, Inc. [3D 16], which was founded by the patent holder. The finished parts of the model are pulled up by the build platform while the liquid is being illuminated by a focused UV light source (most often a laser beam) from below. The light bonds the liquid to a solid, which is cured in conjunction with oxygen that the structure gets exposed to as it is being pulled up by the build plate. The use of light beams to define the structure allows for much greater detail and a vast improvement in speed, in comparison to motors and an extruder depositing the material.

2.2 Further Printing Setups

In comparison to plastics, other substances require a different setup for printing, but the basic principle remains the same - a printing material is deposited one layer at a time, while being cured to create the model.

The company Total Kustom LLC founded by Andrey Rudenko [Rud16] is experimenting with printing concrete, which would make it possible to create whole buildings or parts for prefabricated homes in ways ordinary manufacturing techniques would not allow for. The website shows a few of the projects that have been produced by the company, including a small castle and a hotel suite, complete with two bedrooms, living room, and a whirlpool.

Metals can also be printed, but in this case the layering and curing process differs slightly from how it was described until this point. Some commercial- and industrial-grade printers, which utilize metal as printing material use a process called *sintering*, by depositing a fine powdered metal alloy on the entire printing surface and then melting the areas that need to become solid with a laser beam. This is repeated until the stacking of layers has reached the intended height of the object. Afterwards the model may need to cool down to room temperature, then the excess metal powder can be removed and will be re-used for the next print.

By carefully manipulating the parameters of the sintering process and the selection of a specific source material mixture some remarkable properties can be achieved in the resulting artefact. Traini et al. in their work on laser metal sintering to produce dental implants [TMS+08] outline how this approach can be used to produce implants that can be fixed to the jaw bone much more effectively than comparable implants created using conventional methods.

It is worth mentioning that the field of 3D-fabrication is a rapidly evolving area of engineering. Techniques and machines are permanently improved upon and new approaches and materials are being tested constantly. This thesis therefore can only give a glimpse at the current state of affairs and some aspects might become obsolete relatively fast.

CHAPTER 3

Methods

This chapter will describe how the parts of this thesis project were designed and implemented as well as how the processes come together to form a pipeline for transforming math terms into a virtual 3D-model that is used to provide instructions to a 3D-fabricator in order to end up with a real object.

The Section 3.1 is about the software application that was written during the project to generate the STL files from a parameterized description of the selected object out of a predefined set of objects. In the Section 3.2 and Section 3.4 a short introduction into the intermediate file formats that are required while stepping through the pipeline is provided. Another kind of application is introduced in Section 3.3, which is used to prepare the necessary data for the printing process. Lastly, Section 3.5 presents methods to improve the final result of the print.

3.1 Parameterization Software

In order to generate the dataset for 3D-fabrication of objects from their parameterized representation, a software program was written to preview and tune the parameters and export the specific state of the model using the STL file format. This is then used as the input file for a slicing application to produce the G-Code instructions for controlling the printer hardware and its various settings.

The application consists of a Graphical User Interface (GUI) written using the Qt-Toolkit and is implemented in C++ (see Figure 3.1). At the top right the desired model can be selected which produces a rendering of the object for use as a preview. The right column provides the user with interface elements to modify the available parameters of the model, depending on what model is currently selected. The 3D-mesh is recalculated after each operation on the model, in order to update the preview. The preview can also be rotated and zoomed by interacting with it using the mouse and keyboard. Once the

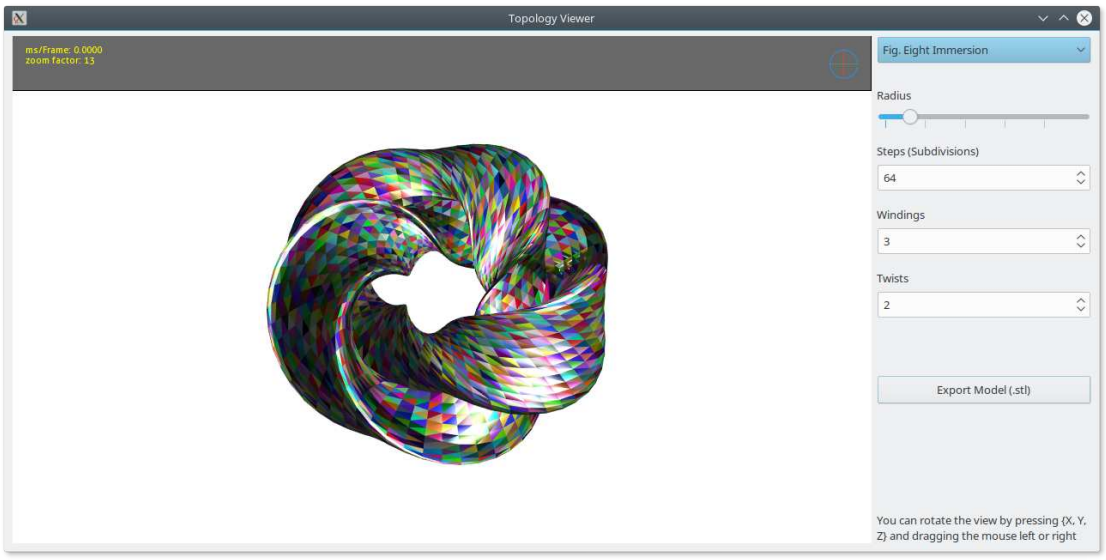


Figure 3.1: Screenshot of the application that was written as part of this thesis. It lets the user set the parameters of the selected model and exports the mesh as STL file.

surface mesh of the model has been calculated from the selected parameters, interaction is provided by manipulation of a view vector and a rotation matrix. The rendering process is out-sourced to the graphics hardware of the computer. This is where the speed of C++ in combination with OpenGL can be observed - resulting in a nice user experience without long response times during the rendering procedure.

Once all the parameters have been set as desired, the resulting mesh can be exported using the "Export Model (.stl)" button at the bottom right. This will produce the STL file which provides the base for all additional processing in order to create an actual "3D-printed" object.

The data is generated by iterating over the fixed parameter space (spherical coordinates θ and ϕ) and generating the X, Y and Z coordinates of each surface point with the variable arguments, set by the user via the graphical interface, inserted into the calculations. Apart from the parameters directly influencing the model, the user is also able to specify the density of interpolation within the bounds of the parameter space. The effect of the subdivision setting is illustrated in Figure 1.1 and this parameter is not automatically decided by the application - the user needs to specify a value that will suit the geometry of the model and the intended purpose of the print. Too few subdivisions and the model will be distorted and many features of the surface will be lost, too many subdivisions will bring no gains to the model quality and just increase export file size and rendering times.

From the generated 3D-model data, the vertices and faces of the triangulated version is calculated, including the surface normal vectors. This virtual surface representation is handed over to the Graphics API *OpenGL* for hardware-accelerated rendering of the

```
solid object_name
  facet normal n_x n_y n_z
    outer loop
      vertex x_1 y_1 z_1
      vertex x_2 y_2 z_2
      vertex x_3 y_3 z_3
    endloop
  endfacet
  [...]
endsolid object_name
```

Figure 3.2: STL file format example. This listing shows a minimal example of a plain text STL file, with one triangle and a corresponding normal vector.

model on the computer screen as well as for exporting the model into an STL file.

3.2 STL File Format

The Stereo Lithography (STL) file format was first specified by the company 3D Systems, Inc. in 1988 [3D 88] and currently still is the predominant format for defining 3D-objects in the fields of CAD and rapid prototyping, especially for exchanging data between applications. Its purpose is to describe the surface of a 3D-object, though, there is no additional information attached to the data, like material, color, or other attributes of the model.

The surface is divided into triangles, each is supplemented with a corresponding normal vector for orientation, coming together to define the surface of the object (see Figure 3.2). The 3D-points and 3D-vectors used are embedded into a Cartesian coordinate systems with arbitrary scale and units. The only restriction for the data specified by the file format is that the coordinate values have to be positive numbers (although, most programs capable of reading the STL file format also handle negative coordinate values correctly). A number used in an STL file needs to be represented in the form: *Sign* - *Mantissa* - `e` - *Sign* - *Exponent*, for example "`2.678e-12`". The file format not only has an ASCII (plain text) representation, but also a binary variant, the latter being more efficient to handle large and/or complex objects consisting of thousands or even more triangles.

3.3 Slicing Software

A slicing software or "Slicer" is necessary to produce the G-Code file (see Section 3.4) from the exported STL file that will be sent to a specific fabrication device. This application will produce the tool path for the plastic extruder and lets the user specify a range of parameters that are needed for controlling the fabrication procedure. This includes extrusion temperature, material flow speed, motor speed, and cooling fan speeds. These

parameters depend on the configuration of the specific printer and may need to be encoded in a specific format, which may vary by manufacturer and model.

As already mentioned in Section 1.2, the slicing tool can also perform some repairs on the imported model, if necessary (see Figure 3.3). These fixes can include flipping the surface normal vectors to clearly define which parts of the model “inside” vs. “outside. Another precondition that is checked by the slicing software that is necessary before printing can take place is that the surface of the model is closed (”watertight“), which will be corrected if that is not the case. Neither of the mentioned steps the slicer runs through to verify the model can be printed should be taken for granted as a solution to all problems, as the repair algorithms can only fall back to heuristics when attempting to fix a broken model. It should always be a priority to already produce a proper model which meets all requirements and needs no further processing to be printable.

Usually the resulting G-Code file produced by the slicer is written to a memory card or USB flash drive, which is then plugged directly into the printer to load the instructions. Newer models of consumer-level printers can also be directly connected to the computer running the slicing software. The printer is then controlled by the software and the current state of the print job can be monitored via the slicing application.

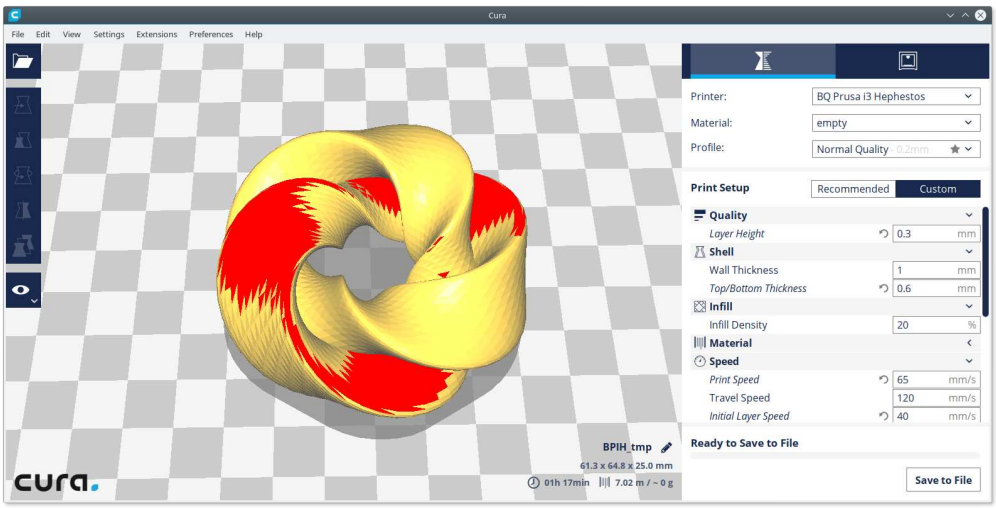
3.4 G-Code File Format

G-Code files are used to instruct computerized machines in the area of Computer Numeric Control (CNC) fabrication, which includes milling, plotting, laser cutting, or - in our case - 3D-additive manufacturing. Its current standardized form is defined by ISO 6983 [ISO09].

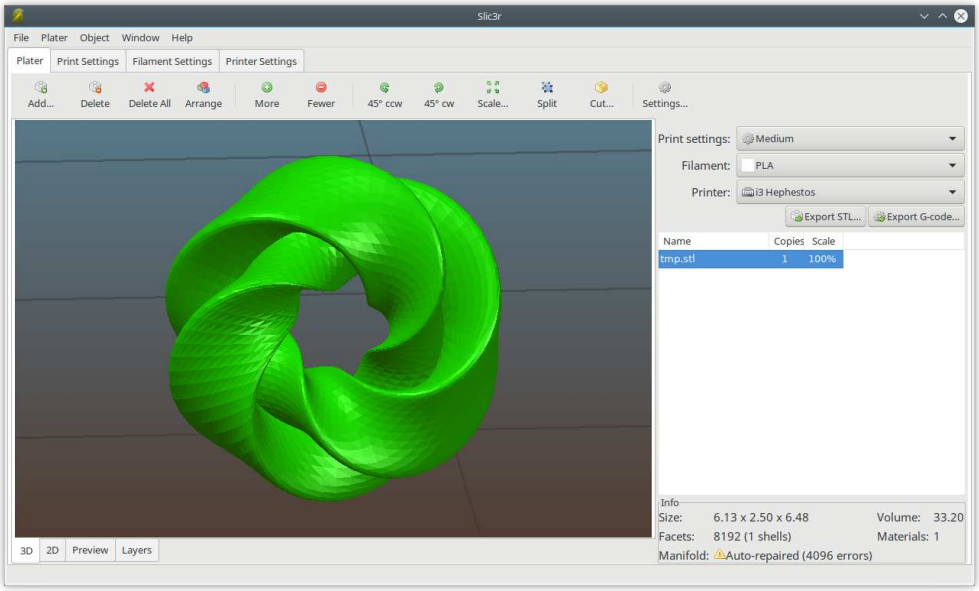
In general, the format is meant to be read by microcontrollers in charge of steering the motors and other attached equipment of a machine, thus the content does not include any information that would have to be processed extensively. Rather than a full object description, there are mostly abstract instructions with a special meaning for the device that will be controlled, which can be directly turned into signals used to actuate the parts of the machine. The interpretation of commands may differ by implementation, in respect of what features a device supports and how the firmware handles them.

The preparatory codes start with the letter “**G**”, followed by a number indicating the desired function according to the firmware specification. Other auxiliary commands may start with the letter “**M**” and address machine-specific commands. Most remaining letters of the (English) alphabet also have a corresponding meaning in G-Code, most importantly X, Y, and Z, which control the tool positioning along the respective axes. Figure 3.4 is a listing showing a very reduced example of a G-Code file, including some commands used in 3D-fabrication.

The flexibility of the format makes it possible to specify some additional notable intricacies, in comparison to the STL format. Material properties like thickness, extrusion rate, and color (if the machine has that capability) can be encoded to be used when producing the model. Commands that influence extrusion and movement speed as well as layer height



(a) Screenshot of the slicing software "Cura" published by Ultimaker B.V. [Ult22]. The application was loaded with a model with broken surface normal vectors (colored in red).



(b) Screenshot of the open-source application "Slic3r" [Sli22], loaded with the same model as in Fig. 3.3a. In the right bottom area there is a message stating the program has auto-repaired some errors.

Figure 3.3: Screenshots showcasing various slicing applications.

```
G92 E0              ; SetPosition Extruder(0)
G28                 ; MoveToOrigin (AllAxes)
G1 F1500            ; Move Feedrate(1500)
G1 X2.0 Y2.0 F3000  ; Move X(2.0) Y(2.0) Feedrate(3000)
G1 X3.0 Y3.0 E22.4  ; Move X(3.0) Y(3.0) Extrude(22.4)
```

Figure 3.4: G-Code file format example. Text after the ";" is considered a comment and is ignored. Source: RepRap Wiki [Rep17]

might have an impact on how well fine details can be reproduced. Changing the nozzle temperature might be required if the desired model should consist of different variations of plastic or of a mixed material with other thermal characteristics. To sum up, the STL file format solely describes the surface geometry of the object, but the G-Code format includes all relevant information that is needed during the production process of a solid model using a 3D-fabricator.

3.5 Printed Model Cleanup

Once the printing process is finished, it may be necessary to remove support structures that were placed below overhanging parts of the model as a scaffold. For thermoplastic printing substrate, this is done either by hand, with a knife and a pair of small clippers or other tools necessary to separate the excess material from the desired artefact.

Another step to enhance the appearance of the object is to smoothen the surface by the removal of all marks of dispensable material left behind or possibly all visible lines that might have been the result of the additive layering process of the printing material. The severeness of these layer-lines will depend on the printing-nozzle diameter and the layer height that is chosen for the printing cycle or might be a result of the limited capability of the printing setup in relation to the intricacies of the 3D-model. The removal may be approached with sandpaper, small files, or a rotary tool and a suitable attachment.

One more noteworthy possibility to flatten relatively small imperfections is to use a chemical approach, which is possible with some types of plastic like ABS. This process involves temporarily placing the model into an air-tight enclosure and releasing acetone vapors inside, which dissolve the outermost layer of the printed artefact. This enables the surface to become supple enough to be susceptible to surface tension, thereby flattening small unevennesses in the material. Once the source of the acetone vapors is removed, the ABS plastic becomes rigid again, as the rest of the solvent dissipates.

CHAPTER 4

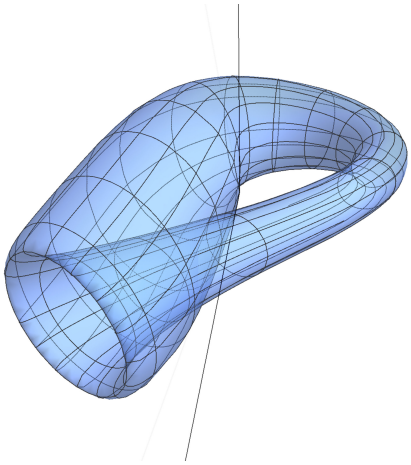
Results and Discussion

This Chapter of the thesis presents the outcomes of applying the methods pointed out in the previous chapter. There is also a part discussing mentionable findings or unexpected aspects of the outlined approaches in regard to the motivation of this thesis. As conclusion there is a list of possibilities on how to expand on the subject of parametric model generation in the area of 3D-fabrication methods.

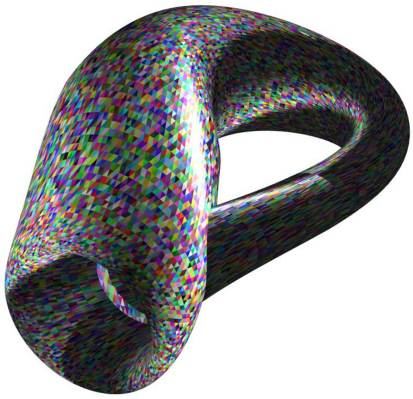
4.1 Outcomes

The goal of this thesis project was to produce a real object from an abstract representation given as a mathematical term. This was achieved by implementing a proof-of-concept software program that allows for selecting one out of a given set of objects, which can subsequently be modified by changing the values of the associated parameters. The software generates a virtual 3D-model that is suitable for exporting and use in a slicing program which prepares the model for ultimately sending it to a 3D-printer, while also allowing the user to specify the parameters necessary for the printing process. Once the model is printed (and some post-processing steps have been applied, if necessary), the result is a real 3D-model of an abstract object, to handle tangibly and intuitively in order to aid perception.

This process is illustrated in detail throughout the various intermediate steps by Figures 4.1, 4.2, 4.3 and 4.4. In conclusion, exploring the possibility of using 3D-printing as means to better understand complex topologies is a challenging topic, but this thesis has produced a number of promising 3D-models, that can be fabricated and become real-world tangible items. These artefacts can conceivably be passed around among students as part of a lecture and hopefully aid them in better grasping the mathematical concepts the objects represent.



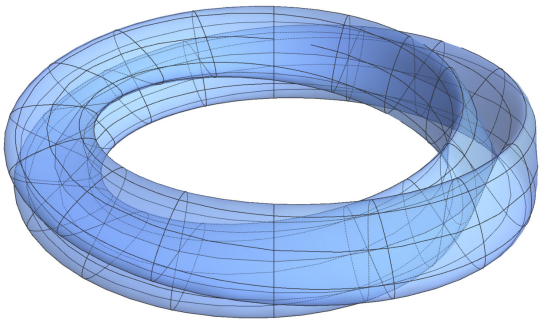
(a) Screenshot of the plotted parameterized form in Mathematica



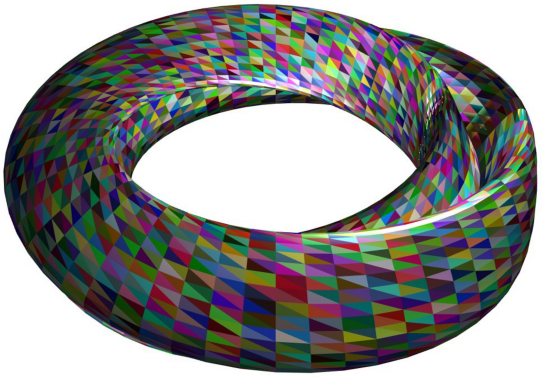
(b) Screenshot of the model generation software

(c) 3D-fabricated object after cleanup using sand paper

Figure 4.1: Comparison of the "Klein Bottle" in virtual and tangible forms.



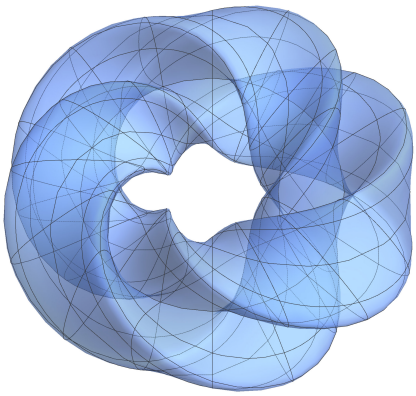
(a) Plot of the model using the same parameters as the application, in Mathematica



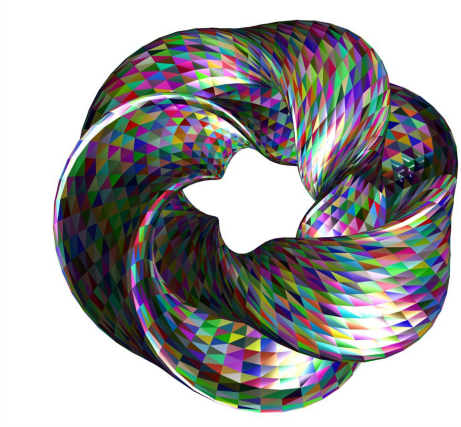
(b) Screenshot of the model as generated by the application

(c) Photograph of the 3D-fabricated object after cleanup

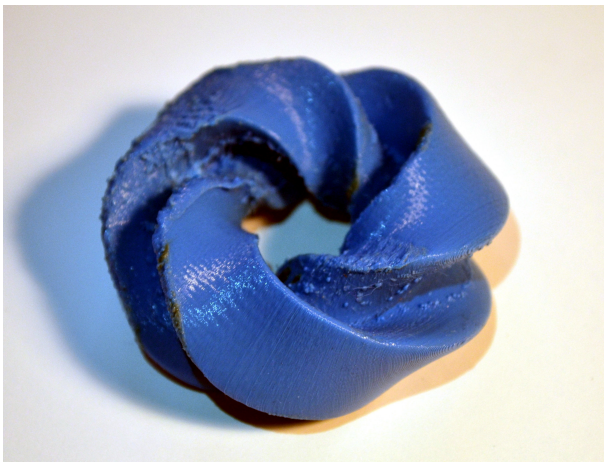
Figure 4.2: Comparison of the "Figure-8 immersion" as it is rendered by different programs and the resulting object.



(a) Plot of the model using Mathematica

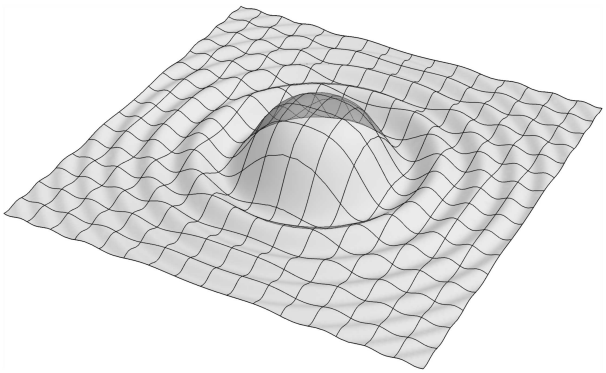


(b) Screenshot of the model as generated by the application

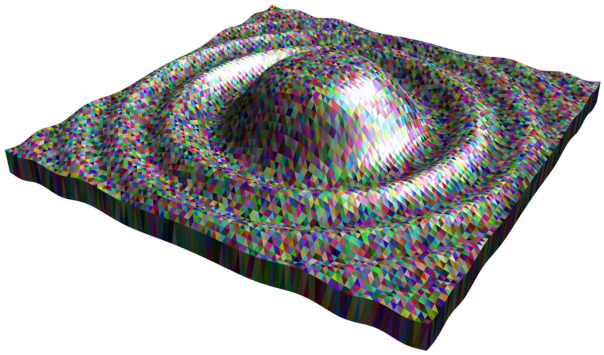


(c) Photograph of the 3D-fabricated object with some remnants of support structures still visible

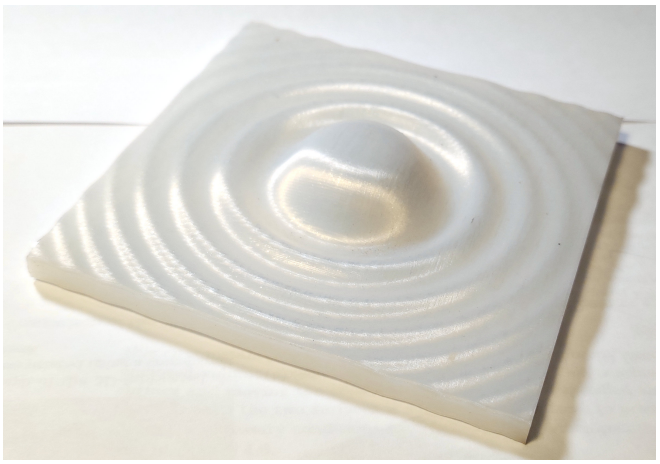
Figure 4.3: Comparison of the "Figure-8 immersion" as in Figure 4.2, with different winding and twist parameters.



(a) Plot of the model using Mathematica



(b) Screenshot of the “Sombrero” model as generated by the application with added brim



(c) Photograph of the 3D-fabricated object which was printed using a semi-transparent plastic filament

Figure 4.4: Comparison of the “Sombrero” model from a plot as a single plane to a 3D-object with a volume, and the resulting printed object.

(a) 3D-model of the Klein Bottle with the gap on the left side.

(b) Photograph of the printed model with the gap visible on the front facing side.

Figure 4.5: Model of the Klein Bottle with an intentional gap in the surface, allowing the slicing application to recalculate surface normal vectors for determining "inside" vs. "outside".

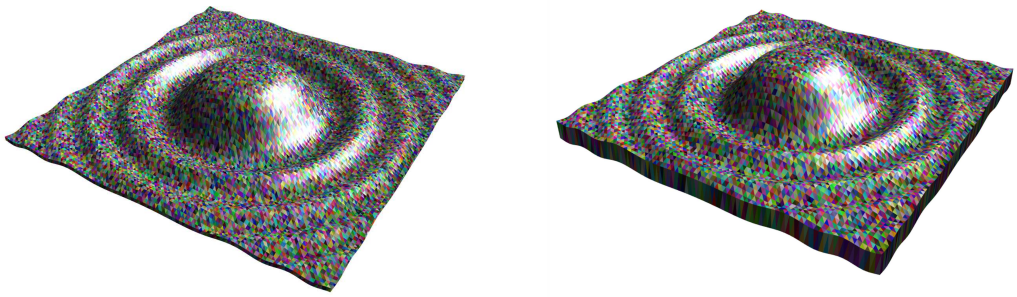
4.2 Problems

During testing/handling of the STL file export it became apparent that most 3D-fabrication software applications perform at least some rudimentary checks on the models that are imported, and, if any inconsistencies were found, they try to repair the internal representation of the model. For instance, it was possible to set all the normal vectors of an object to zero length prior to exporting the STL file, which prompted the software to recalculate the vectors upon selecting the file for use.

This attempt at fixing the model was impossible for geometries that featured self-intersections or normal vectors that could not be explicitly determined. In this case, an artificial "cut" was introduced into the 3D-model to aid the repair mechanism and allow it to handle the model correctly (see Figure 4.5).

The normal vectors need to be consistent for the slicer to determine the "inside" and "outside" of the model, so in the case where the calculation of the surface using the parametric mathematical term does not produce the vectors in the correct orientation, this needs to be fixed. This problem is illustrated in Figure 3.3, where both screenshots of slicing programs display errors in regard to surface normal vectors.

A different problem came up for the decision which models to select for use in the software program written for this thesis with regards to feature size. The question is how to handle models that may have structures smaller than the minimum resolution the 3D-printer is capable of. While an actual solution to this problem is out of scope



(a) 3D-model of the “Sombrero” showing only a single surface.

(b) The same model with a brim around the edge, connecting the duplicated top and bottom surface.

Figure 4.6: The “Sombrero” model displaying the brim that was added to connect the top surface with the copy at the bottom to produce an enclosed volume..

of this thesis, nevertheless this circumstance needed to be handled for the “Sombrero” model. The object has no enclosed volume, so one needed to be generated. This is achieved by duplicating the surface and moving the copied version along the Z-axis by a specified amount, subsequently the brim between the two surfaces is surrounded with faces to produce a closed volume that meets the necessary conditions to be printed. This manipulation of the 3D-model can be seen in Figure 4.6.

Another challenge to keep in mind is the finite accuracy the printing machine can reasonably achieve, depending on the setup of the device and the process used. With current semi-professional Fused Deposition Modeling (FDM) printers the limit of 3D-features that can still be rendered would be around 0.1 mm, anything smaller than that will become altered by aliasing or other unwanted effects to the detriment of the final artefact.

When creating models with very fine-grained details one more aspect that needs to be taken into account is the wall thickness. This is usually a measurement determined by the nozzle size or other factors, depending on the printing process, that also limits the size of features displayed as part of the model. In the same way, layer height and the material used for printing have a big impact on the outcome, and should be adjusted and chosen accordingly.

Often it will be necessary to create a few test runs with different configurations in order to determine the best combination of the aforementioned parameters to produce a result that will be satisfactory.

4.3 Future Work

The software program created as part of this thesis project is mainly a proof-of-concept and provides the basic means to generate 3D-models from certain given mathematical terms. In order to provide a better output for a more generic use, various improvements could be made to optimize the generated data for 3D-fabrication. This could involve, for example, making sure, the volume does not have an excessive amount of curvature or has a certain minimum thickness throughout as the preview does not relate the model size to an actual measurement unit like millimeters. Parts of the object could be challenging to print as outlined in the previous Section 4.2.

Another striking limitation of the application is the fixed set of options and the parameters that can be customized. This could be addressed by providing an input for arbitrary math terms and defining the variables for changing the appearance. Giving the user the ability to input their own models brings with it the additional challenge of providing sensible default values for all possible parameters and limiting the valid range to an interval that can produce meaningful results.

Since printers have gained the ability to deposit different materials or introduce various colors into the final artefact, it could be beneficial to use this functionality to enhance a certain aspect of the model. Some examples might be to highlight a notable feature of the 3D-object in an eye-catching color or using a transparent material to reveal an internal structure and also showing isolines as mentioned in Figure 1.3b.

One last note to improve upon could be the integration with other pre-existing software tools, that are used to create 3D-models (e.g. Blender [Ble22] or OpenSCAD [Ope22]). The software from this thesis could be restructured as a plugin to those other applications and might provide a practical addition to their capabilities. In turn a user might also profit from established user interfaces and 3D-modelling tools.

Some of the insights brought up in this thesis might also be of interest to the creators of slicing software. Most of those tools already provide many options and attempt to optimize a given 3D-model for printing, but particularly the effort of fixing a mesh that appears inconsistent was conflicting with the models that were generated in the context of this project. Handling these sort of border cases would make slicing software even more useful and robust and might be desirable for the software vendors to achieve.

List of Figures

Bibliography

[3D 16] 3D Systems, Inc. 2016. URL: https://www.3dsystems.com/ (visited on 11/04/2016).

[3D 88] 3D Systems, Inc. *StereoLithography Interface Specification*. July 1988.

[Ble22] The Blender Foundation. blender.org - Home of the Blender project - Free and Open 3D Creation Software. 2022. URL: https://www.blender.org/ (visited on 03/16/2022).

[Bou96] Paul Bourke. Klein bottle. 1996. URL: http://paulbourke.net/geometry/klein/ (visited on 01/18/2017).

[GZR+15] Wei Gao, Yunbo Zhang, Devarajan Ramanujan, Karthik Ramani, Yong Chen, Christopher B. Williams, Charlie C.L. Wang, Yung C. Shin, Song Zhang, and Pablo D. Zavattieri. The status, challenges, and future of additive manufacturing in engineering. *Computer-Aided Design*, 69:65–89, 2015. ISSN: 0010-4485. DOI: http://dx.doi.org/10.1016/j.cad.2015.04.001. URL: http://www.sciencedirect.com/science/article/pii/S0010448515000469.

[Hul86] Charles W. Hull. Apparatus for production of three-dimensional objects by stereolithography. (US 4575330). March 1986. URL: http://www.google.com/patents/US4575330 (visited on 07/22/2016).

[ISO09] ISO. Automation systems and integration – Numerical control of machines – Program format and definitions of address words – Part 1: Data format for positioning, line motion and contouring control systems. Standard 6983-1:2009, International Organization for Standardization, Geneva, CH, December 2009. URL: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=34608 (visited on 01/18/2017).

[KLM+10] Olga Karpenko, Wilmot Li, Niloy Mitra, and Maneesh Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, November 2010. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.151.

[Ope22] OpenSCAD project contributors. OpenSCAD - The Programmers Solid 3D CAD Modeller. 2022. URL: https://openscad.org/ (visited on 03/16/2022).

[Rep17] RepRap Wiki contributors. G-code - RepRap Wiki. 2017. URL: http://reprap.org/mediawiki/index.php?title=G-code&oldid=177926 (visited on 01/18/2017).

[Rud16] Andrey Rudenko. Totalkustom, Inc. 2016. URL: http://www.totalkustom.com/ (visited on 11/06/2016).

[Sli22] Slic3r project contributors. Slic3r - Open source 3D printing toolbox. 2022. URL: https://slic3r.org/ (visited on 03/16/2022).

[TMS+08] T. Traini, C. Mangano, R. L. Sammons, F. Mangano, A. Macchi, and A. Piattelli. Direct laser metal sintering as a new approach to fabrication of an isoelastic functionally graded material for manufacture of porous titanium dental implants. *Dental Materials*, 24(11):1525–1533, November 1, 2008. ISSN: 0109-5641. DOI: 10.1016/j.dental.2008.03.029.

[Ult22] Ultimaker B.V. Ultimaker Cura: Powerful, easy-to-use 3D printing software. 2022. URL: https://ultimaker.com/software/ultimaker-cura (visited on 03/16/2022).