UNIVERSITY OF CALIFORNIA
SANTA CRUZ

PHYSICALIZING VIRTUAL MODELS CREATED BY PHYSARUM POLYCEPHALUM 3D SIMULATION

A masters thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTATIONAL MEDIA

by

Drew Ehrlich

June 2023

The masters thesis of Drew Ehrlich
is approved:

Professor Angus Forbes, Chair

Professor Sri Kurniawan

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

Abstract

Physicalizing virtual models created by Physarum polycephalum 3D simulation

by

Drew Ehrlich

This thesis presents a pipeline for creating visually compelling bio-inspired versions of triangle meshes that can be 3D printed using consumer hardware. The process to make these models uses the simulation software PolyPhy, whose behavior is governed by an algorithm that mimics the optimal foraging behavior of the Physarum polycephalum slime mold. The structures created with this technique can both serve as computational art and also lay the foundations creating novel infill structures for filament based 3D printing.

Acknowledgments

This thesis is dedicated to everyone who has supported me through my journey into the world of academic research. First and foremost, I want to thank Dr. Oskar Elek for directly supervising and supporting the work shown in this thesis, along with creating the software that allows this work to be done in the first place.

This work would also not be possible without Professor Angus Forbes. Angus took a chance on me as a undergrad with little research experience and unimpressive grades and allowed me to learn the ropes as a undergraduate researcher and eventually become a graduate student under his supervision.

I would also like to thank Professor Mircea Teodorescu and Dr. Yohei Rosen for giving me an opportunity to work in the Braingeneers group, and financially supporting me through that work. Being a researcher in this group has allowed me to explore my interests in electronics and hardware in a supportive environment while still being a student in the Computational Media department.

Lastly, but most certainly not least, I would like to thank Professor Sri Kurniawan for becoming my academic advisor as I move from the Masters program to pursuing a PhD in Computational Media starting this year.

Chapter 1

Conceptual Foundations

Portions of this thesis pull material from the extended abstract "Scaffolding Generation using a 3D Physarum Polycephalum Simulation" which was presented during the poster session at the ACM Symposium on Computational Fabrication in Seattle, Washington on October 27, 2022 [1].

1.1 Introduction

Humans have always been interested in patterns, geometric or otherwise. Archaeological evidence of this propensity can be found all over the world and can date back thousands of years [2, 3]. Generative design and biologically inspired design are a modern extension of this practice, using modern capabilities to create novel visual media [4].

Biomimetic design is an area of biologically inspired design dedicated to mimicking natural processes and systems. Whether it is in the design of airplane wings to match the aerodynamics of birds [5], or of buildings to mitigate the effects of climate change [6], both flora and fauna can serve as inspiration for approaching design problems. One of the organisms of interest in biomimetics is the Physarum polycephalum, a bright yellow slime mold that forages for resources in a way that mimics conventional solutions to shortest paths problems. Because of this property it has also been a subject of virtual visualization experiments, and makes it unique from parametrically defined optimization algorithms.

The open source software PolyPhy builds on these previous works by creating a three dimensional Physarum polycephalum inspired simulation. However, the structures created with PolyPhy previously could not be converted into physical objects.

This thesis presents a technique that modifies PolyPhy to output intricate three dimensional networks that can be 3D printed with any technique. The following will be presented to demonstrate this contribution:

- A technical description of the process used to create these objects.
- Strategies for increasing quality in the printed objects.
- A sample of resulting 3D printed objects.

1.2 Background

1.2.1 Physarum polycephalum

Physarum polycephalum is an acellular slime mold that is commonly bright yellow in color. The way that Physarum polycephalum searches for nutrients has found to be of scientific interest, with researchers discovering that it solve mazes and shortest paths problems [7, 8, 9, 10]. This optimizing property has been used in computing to generate complex 2D simulation systems in works done by Adamantzky and Jones, with artists like Sage Jenson using it to create generative artworks [11, 12, 13].

At this point, while Physarum polycephalum is somewhat obscure, it has managed to capture global attention twice in the past decade [Figure 1.1], both relating to an exhibit featuring the slime at the Paris Zoological Park in France [14, 15]. Since Physarum polycephalum is a well studied natural organism that can solve shortest paths problems, the public could benefit from observing its behavior to better understand these kinds of problems and the way they are solved in nature. Physarum polycephalum can also be found all over the world, making it accessible for observation.

1.2.2 Structure Optimization Algorithms for Meshes

In virtual object modeling, meshes are three dimensional sets of vertices, faces, and edges that form the shape of a greater object. The shape of the faces that make up

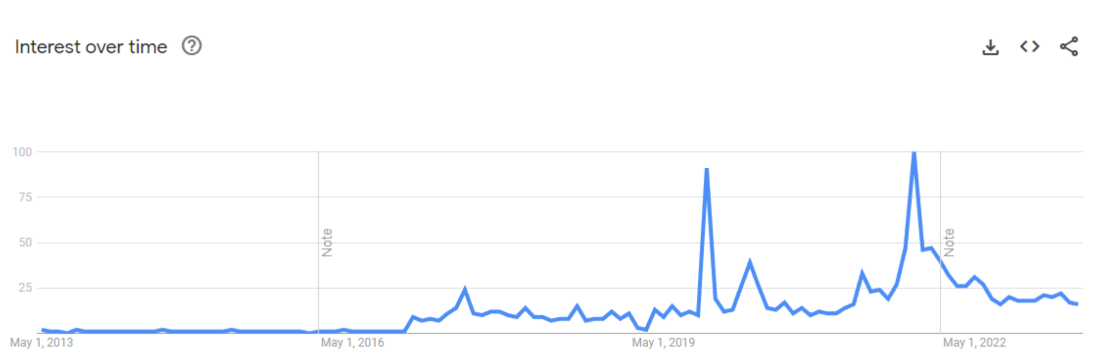

Figure 1.1: A Google Trends chart showing the two spikes in worldwide searches for Physarum polycephalum between May 2013 and May 2023.

the objects presented in this thesis are triangles. Any 3D model seen in digital media, whether it is for a video game or 3D printing, is a mesh.

Finding the optimal structure that requires a minimum amount of material for these meshes has long been an interest in computer graphics. Two areas in particular are relevant to the technique presented in this thesis: topology optimization and scaffolding construction.

Optimizing the topology of a mesh has often been accomplished using parametric algorithms. These algorithms take a set of points and optimize the surface. Often these are based on fundamental geometric tessellations like Voronoi, Delaunay, gyroid, and their higher-dimensional extensions [17].

In terms of creating scaffolding structures, one of the more common applications has been creating infill structures for 3D printed objects. For example, the optimization of both interior and exterior scaffolding structures has been extensively studied in optimizing infill and supports [18, 19].

1.2.3 Polyphorm and PolyPhy

Polyphorm and PolyPhy are a pair of physarum polycephalum inspired 3D visualization and simulation softwares, of which Polyphorm is the predecessor. Both softwares are built on the Monte Carlo Physarum Machine (MCPM), a generative algorithm that creates optimal tansport networks between points in a 3D environment. The

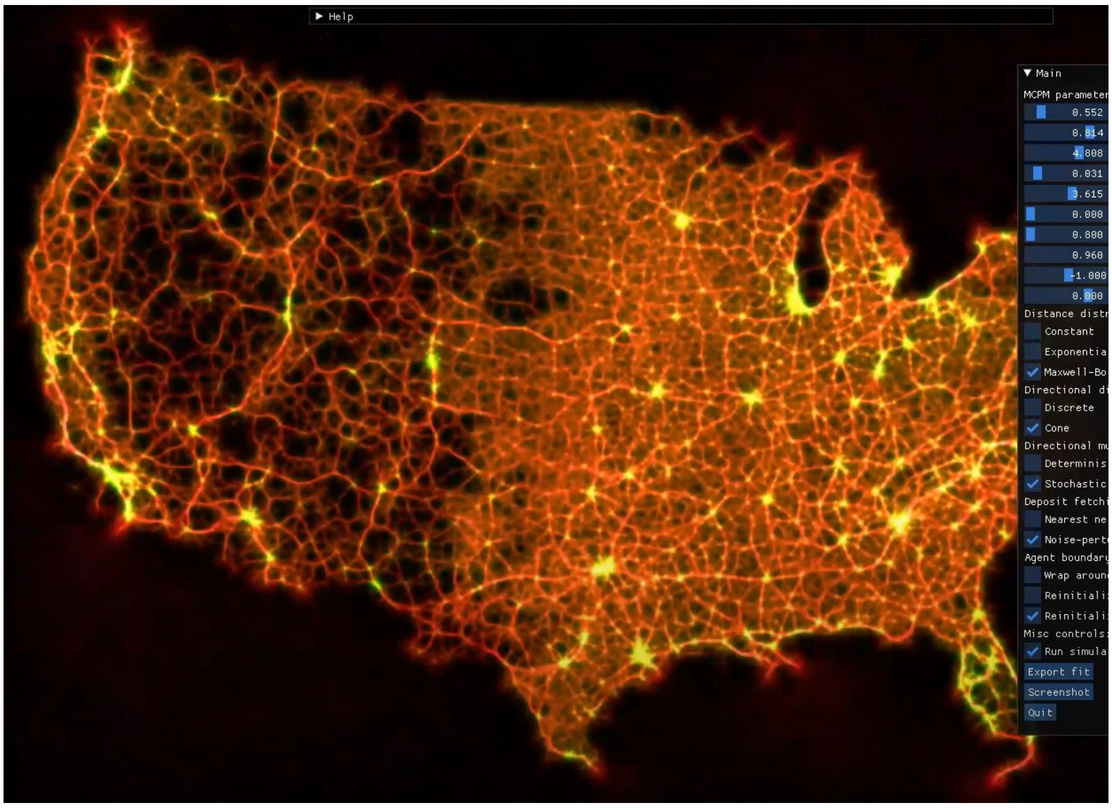

Figure 1.2: A map showing how PolyPhy's simulated Physarum polycephalum agents would have designed the highway system of the United States of America [16].

actions of this algorithm mimic the behavior of the Physarum Polycephalum, and the MCPM and Polyphorm were published together in 2022 [23, 24]. Polyphorm was created with astronomers to create a novel visualization technique to accurately model 3D cosmological datasets [25]. While Polyphorm is technically open source, it has both a Windows dependency and a very unintuitive intial compilation process for non-technical users.

PolyPhy is the second version of Polyphorm – it still uses the MCPM to define its behavior, but it was designed from the outset to promote open source collaboration and run in a Python environment. PolyPhy is not only is easier to use than Polyphorm, but it also encompasses more modalities – Polyphorm can only work with 2D data and meshes, while PolyPhy can also create transport networks from voxels. Development is currently in progress with the support of the Open Source Program Office at the

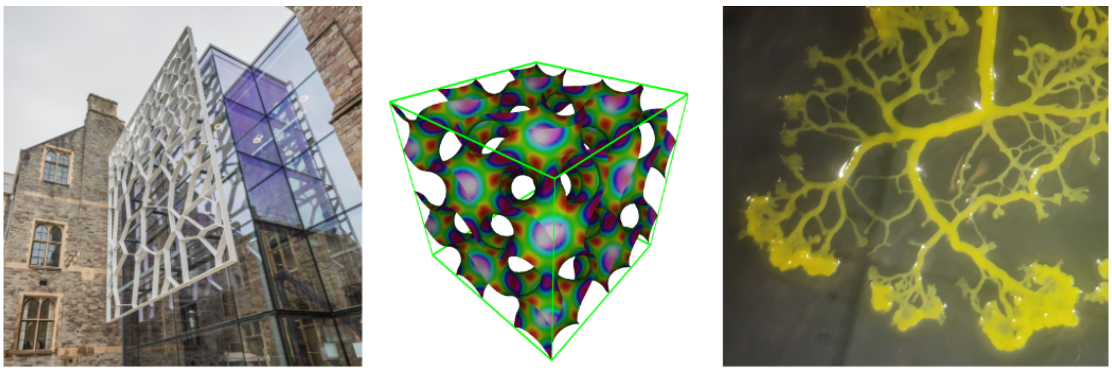

Figure 1.3: Examples of optimization models in action: Voronoi (left) [20], gyroid (middle) [21], and the Physarum Polycephalum (right) [22].

University of California, Santa Cruz. The latest version of the software can be found at https://github.com/PolyPhyHub/PolyPhy.

1.2.4 How Does PolyPhy Work?

The simulations created by PolyPhy software consist of three entities: the input data, the agents, and the trace. The agents are the dynamic building blocks of the MCPM algorithm, which guides them to connect 'food sources' (the data) in an optimal way. The data is an unordered set of points in 3D space. For example, the cosmological datasets used by the authors of the original Polyphorm paper [25] are be the kind of data that would be loaded in to PolyPhy. In this case, the input data is all of the vertices on the triangles of the input voxelized mesh (see Section 2.1.1). The trace is a 3D histogram of agent trajectories averaged over the runtime of the simulation as they navigate between the data points. The final printed objects are physical representations of the networks the trace forms in the simulation, and individual segments and branches of this network are called filaments (term adopted from the cosmological nomenclature). The process of changing parameters in PolyPhy to achieve a desired visual result is called fitting.

In short, the agents seek out the data and leave behind the trace, which latently encodes the shape of the final object or design to be printed. Most of the work in fine tuning structures in PolyPhy consists of manipulating the properties of the agents

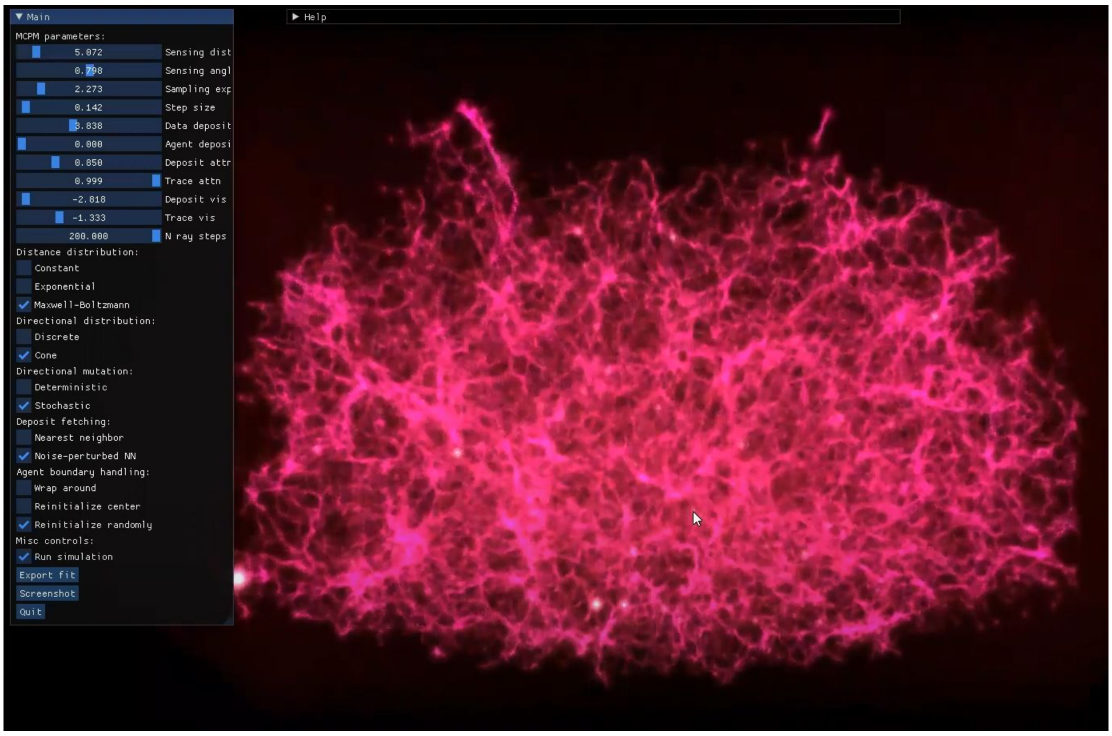
```
▼ Main
MCPM parameters:
      5.072        Sensing dist
      0.798        Sensing angl
      2.273        Sampling exp
      0.142        Step size
      3.838        Data deposit
      0.000        Agent deposi
      0.850        Deposit attr
      0.999        Trace attn
     -2.818        Deposit vis
     -1.333        Trace vis
    200.000        N ray steps
Distance distribution:
[ ] Constant
[ ] Exponential
[✓] Maxwell-Boltzmann
Directional distribution:
[ ] Discrete
[✓] Cone
Directional mutation:
[ ] Deterministic
[✓] Stochastic
Deposit fetching:
[ ] Nearest neighbor
[✓] Noise-perturbed NN
Agent boundary handling:
[ ] Wrap around
[ ] Reinitialize center
[✓] Reinitialize randomly
Misc controls:
[✓] Run simulation
```

Figure 1.4: A screenshot of the user interface in PolyPhy. The sliders on the left show the parameters of the simulation that can be modified. Reprinted with permission from Dr. Elek.

to produce a more desirable trace. To improve the printability, visual appeal, and structural integrity of the final models, good quality fits were seen to have clear discrete filaments that did not leave large quantities of agents floating in between. Targeting these features creates objects that are easier to see inside and have thicker, less fragile features (more details in Section 2.1.4).

1.2.5 Filament Based 3D Printing and Infill Structures

Fused Deposition Modeling (FDM) is a commonly used 3D printing technique for consumer level machines. In FDM, material from a wound spool of cylindrical plastic that is a few millimeters thick is pulled through a heated block of metal and extruded through a small nozzle with an opening that is most commonly .4 millimeters across that has been heated to upwards of 200 degrees Celsius. A diagram of this process can

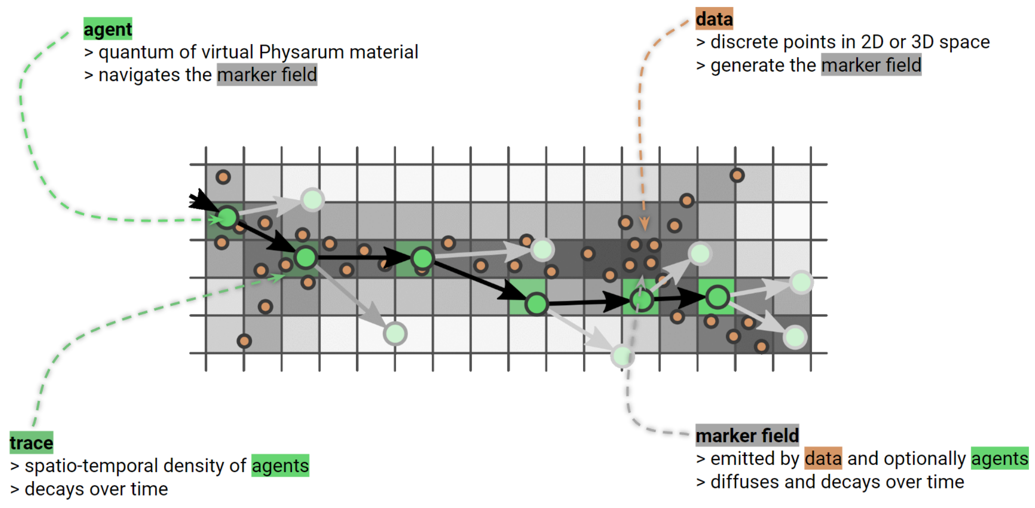
agent
> quantum of virtual Physarum material
> navigates the marker field
data
> discrete points in 2D or 3D space
> generate the marker field
trace
> spatio-temporal density of agents
> decays over time
marker field
> emitted by data and optionally agents
> diffuses and decays over time

Figure 1.5: An annotated figure showing the way an agent moves through PolyPhy's simulation. Reprinted with permission from Dr. Elek.

be seen in Figure 1.6. This process of melting plastic and forcing it through a small hole allows for rendering fine detail in a printed object.

Infill structures are internal scaffoldings that allow printed objects to remain intact while lowering the amount of plastic required for each print job. While a printed object could be solid on the inside, using infill structures will lead to the same visual result with less material used. These patterns can range from arrays of rectangles to more complicated patterns like gyroid. A model with the infill exposed can be seen in Figure 1.6.

1.2.6 Previous Work

The first version of the mesh reconstruction pipeline presented in this thesis was the author's undergraduate thesis project. This earlier version of the project used the same implementation of the Marching Cubes reconstruction algorithm from Python library pymcubes, but could only represent the outer shell of the mesh which resulted in hollow objects [28].

However, this previous approach had a few major weaknesses:

- When 3D meshes were used with this earlier pipeline, the results would all be

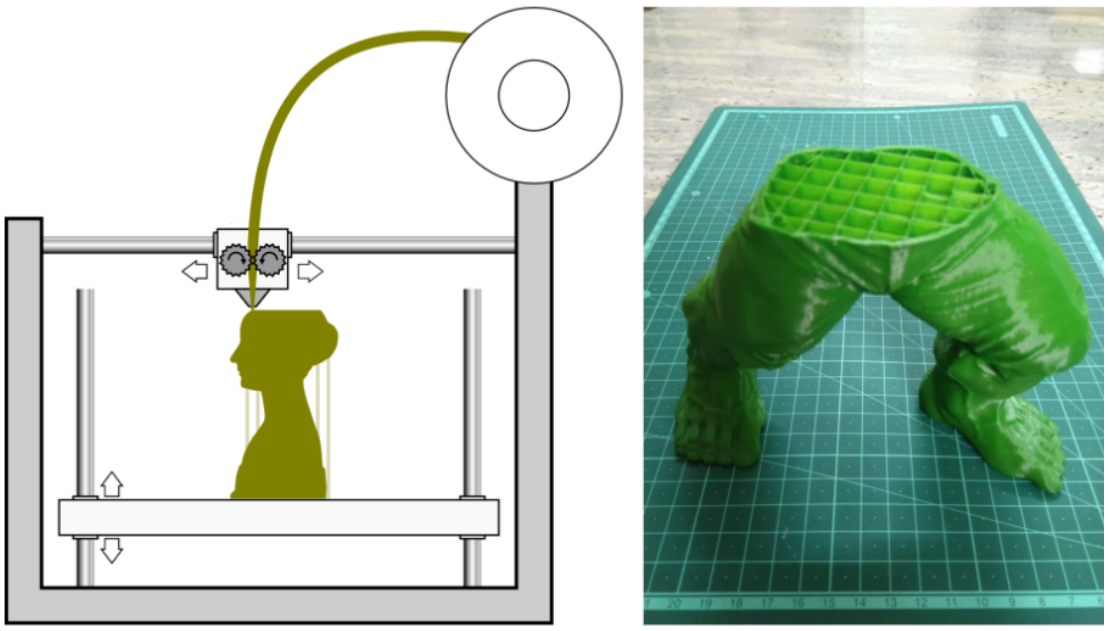

Figure 1.6: A graphic representation of the FDM process [26]. Plastic is forced through an extruder and and a nozzle onto the printed object (left). A half printed object showing the rectangle based rectilinear infill structure (right) [27].

hollow, since the only data points in the simulation were on the surface of the mesh.

- The only way to get models with structure on the inside was to artificially insert points into the interior of the mesh randomly, which is not true to the object's actual interior structure.

These compromises were results of not using voxelized meshes and not modifying the kernel of the program to handle these data. While the resulting printed objects showed promise as a proof of concept, the overhaul of the pipeline into what is presented in this thesis fully achieves the goals set out by this earlier iteration.

1.3 Project Goals

There are three main goals with this project.

1. To create 3D printed objects that are composed of filaments generated by PolyPhy in all parts of the volume.
2. To improve a previously developed technique for creating hollow objects whose shells are composed of filaments generated by Polyphorm.
3. To use the technique developed to explore if the filaments generated in PolyPhy can be used to create a unique and optimal infill structure for filament-based 3D printing.

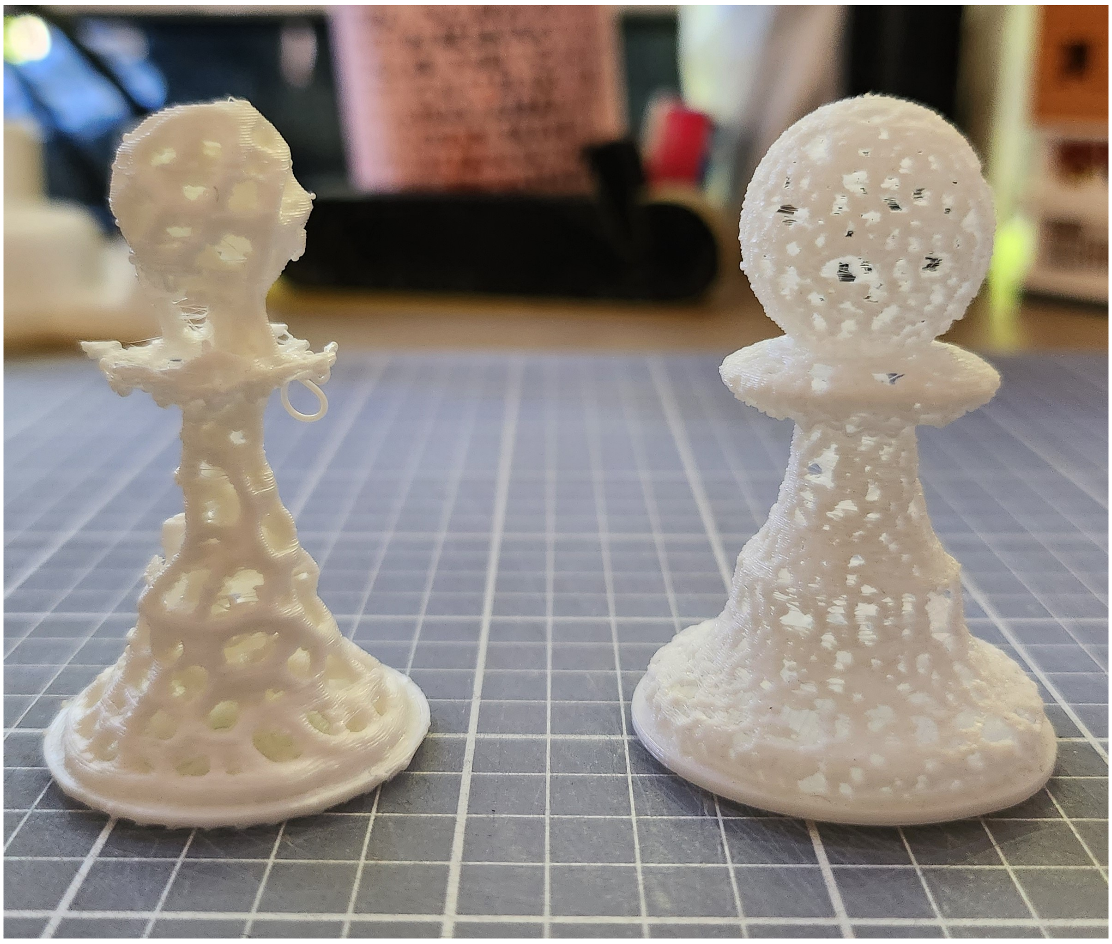

Figure 1.7: A printed version of a pawn from the previous version of this project (left), and a print of the same original model being passed through the current version of the pipeline (right).

Chapter 2

Implementation and Applications

2.1 Mesh Generation Pipeline

In its current form, passing a mesh through PolyPhy uses a standalone Jupyter Notebook for all steps of the process. However in the future this process will be integrated into a unified library of PolyPhy utilities.

For meshes to use this pipeline successfully, they must be manifold as part of this process involves defining the shape of the interior of the mesh. Thus, non-manifold meshes cannot be used with PolyPhy in this way.

As of writing, PolyPhy runs best on graphics cards using a CUDA backend, so users intending to try this pipeline for themselves should use a computer with a NVIDIA GPU with at least 1GB of video memory at a bare minimum. Using a more powerful GPU will reduce processing times and increase frame rates when working inside PolyPhy itself. The meshes generated for this project were created on a desktop with a NVIDIA RTX 2080 Ti, which had PolyPhy running at a framerate of approximately 100 frames per second. Since PolyPhy also creates a graphical user interface (GUI), this pipeline cannot be fully run in a virtual headless environment like Google Colab.

2.1.1 Mesh Preprocessing and Voxelization

This pipeline only supports the loading of STL (Standard Triangle Library) meshes. Once the mesh is loaded, it is voxelized. Voxelization is the process of taking a mesh and converting into a set of voxels (three dimensional pixels). Since meshes are usually hollow, voxelizing a mesh makes it possible to know the full set of points in the 3D space are part of the volume taken up by the mesh.

In this case, voxelization is done using the pyvista Python library which results in an array of all of the voxels that compose the volume of the mesh. This array is called a mask. The mask is then reformatted to be passed into PolyPhy's kernel, which will force the agents in the simulation to stay inside of its bounds.

2.1.2 Fitting in PolyPhy

This mesh generation pipeline runs on a modified version of PolyPhy's 3D simulation kernel. The modifications made consist of changes to the behavior of agents to enforce the boundaries defined by the mask, and the whole pipeline is contained within a Jupyter Notebook environment.

Once PolyPhy is running, fine tuning the simulation can begin once the following settings are applied:

- Data Deposit should be set to 0 since points that make up the mask act as the data in the simulation
- Agent Deposit should be set to its maximum value, since the if it is lower, filaments in the final object will be weaker and thinner.
- Deposit and Trace Attenuations should be set to their median values to allow the agents in the simulation to be able to form solid connections with each other.
- The Step Size parameter controls the speed of the agents in the simulation. The higher the step size, the faster they move – however raising the step size too high will make the simulation more blurry. This effects the final model by sealing off spaces in the objects where it was meant to be open.
- The Sampling Exponent parameter controls how sharp features are. The higher the value is, the sharper the features will be. However, this comes at the cost of reducing the complexity of the resulting structure.

Once these settings are in place, adjust the Sense Angle and Sense Distance parameters. These two parameters directly control the perceptual abilities of the individual agents. Sense Angle changes the field of view for identifying data to move towards in the simulation, and Sense Distance controls how far away each agent 'looks' for data.

These are some points to keep in mind while choosing a Sense Distance:

- If the Sense Distance is too low, the agents will not fit the entire mesh and will instead form clumps of disconnected filaments.
- If the Sense Distance is too high, the agents will miss parts of the mesh since they will be looking too far away.

An important thing to remember when fitting is that the simulation is a living system, so the fit observed will be dependent on the sequence and timing of changed parameters. If an undesirable and unrecoverable state is reached in the simulation, the Step Size parameter can be set to its highest value, and the simulation will dissolve into its original state until the value is lowered.

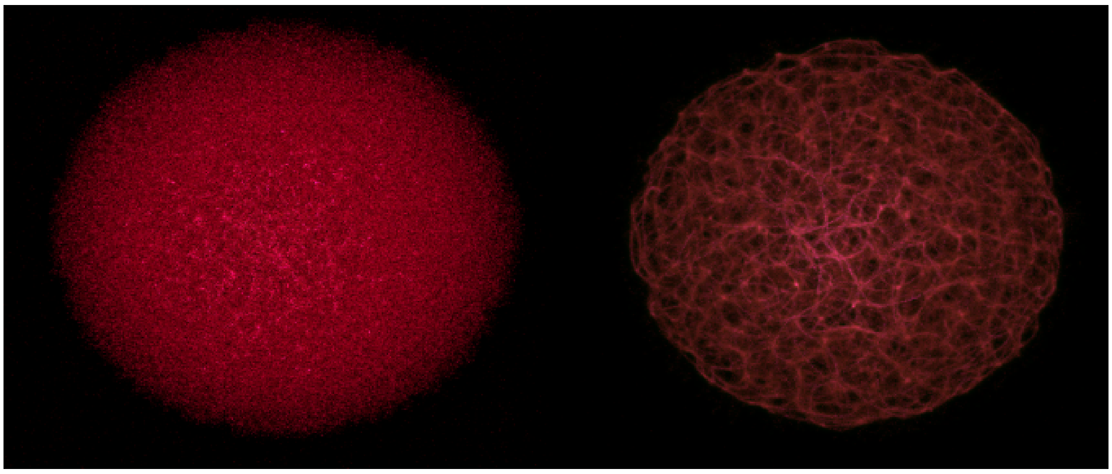
Figure 2.1: A sphere before being fit in PolyPhy (left) and the same sphere after PolyPhy's parameters have been tuned (right).

While fitting, pay attention to how much the fit is decomposing – if left untouched, the filaments created by the simulation will collapse, leaving an abstract and minimalist version of the input mesh. This effect can be reversed by setting the sensing distance to 0, waiting for the fit to completely dissolve, and then returning it to its previous value.

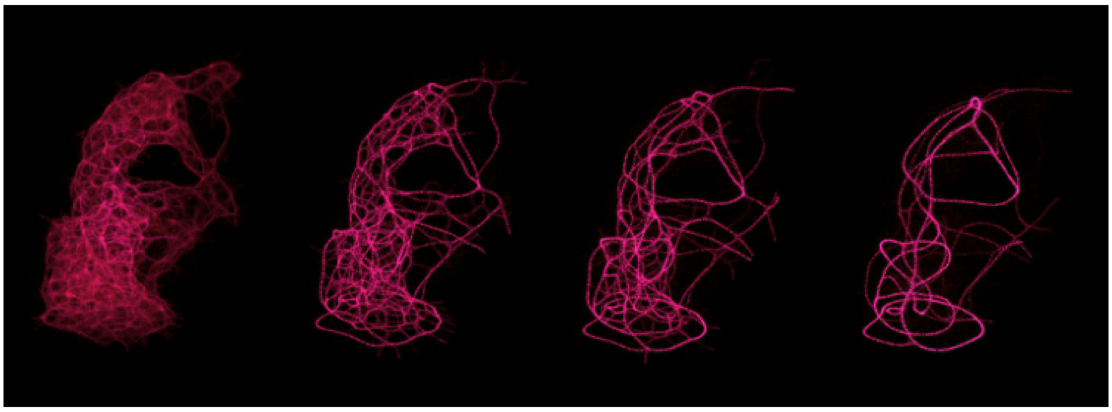

Figure 2.2: The PolyPhy fit of a 3D mesh of Rodin's "The Thinker" sculpture collapsing over time [29].

Once a satisfactory fit has been created, clicking the 'Export Fit' button at the bottom of the slider interface will store the current state of the simulation. As a result, the simulation will freeze while the export is taking place. Once the export finishes, the simulation will resume. When creating meshes with this pipeline, do not to export multiple fits in one session since the pipeline is design to reconstruct the most recent mesh exported from the software.

2.1.3 Export and Reconstruction

Once the fit has been exported, it is stored as a numpy array (.npy file) in the /data/fits directory of the local PolyPhy installation. Part of the export process is defining a pointer to the newly created file, so no additional work needs to be done.

The first step in postprocessing is to apply a slight gaussian blur to the exported fit to make the end result smoother, which leads to higher quality printed objects. The mesh is then passed through the implementation of the Marching Cubes algorithm found in the pymcubes Python library. The original Marching Cubes algorithm was originally published by Lorenson and Cline [30]. The result of this reconstruction is then converted from the Collada (.dae) format that the library outputs and converts it to an STL. The resulting STL is then decimated to make the filesize smaller at minimal cost to surface resolution. Models can commonly exceed millions of triangles, so this decimation allows for easier dissemination and minimizes computation required by 3D printer slicing softwares, which convert 3D models into 3D printer machine instructions.

All 3D models passed through PolyPhy featured in this document are available at this link: https://tinyurl.com/thesismodels.

The Jupyter notebook used to create the models featured in this document are available at https://tinyurl.com/polyphynotebook.

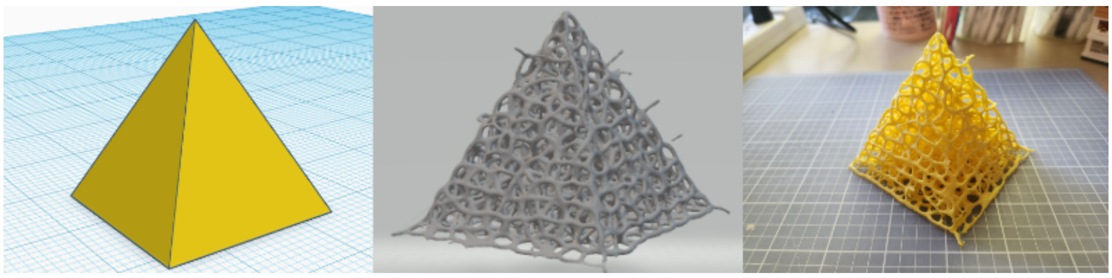

Figure 2.3: A mesh of a pyramid (left), the mesh after being put through PolyPhy (middle), and the printed version of that result (right).

2.1.4 Printing

All printed objects presented in this thesis were sliced using Ultimaker Cura 4.11.0 and printed on a Creality Ender 3 that has been converted to use direct drive extrusion.

There are a few methods for promoting higher quality printed objects.

1. Printed objects should have dimensions no smaller than 50 millimeters. Scaling the filaments created by the simulation smaller than this will cause them to be to small to render properly in plastic, and can lead to fragile and unstable objects.

2. Use support blockers to stop support generation for the upper ninety percent of the mesh. If not enabled, the entire structure will be filled with support which will create objects where the support structures cannot be removed. The structures generated with this technique are self-supporting in nature so the only support necessary are to help the bottom of the model adhere to the 3D printer's build platform.

2.2 Results

The models presented in this thesis were all printed on the printer described in the previous section using polylactic acid (PLA) filament from various suppliers. All images were taken with a Samsung Galaxy Z Flip 4 cell phone.

The printed objects shown in this thesis can be split into three categories: shape primitives, existing 3D models, and experimental designs. The shape primitives are meshes of basic shapes run though the pipeline as examples to benchmark with - any technique like this one needs to be able to work with simple models. These stand in contrast with the existing 3D models which show the extents that this technique can be applied. Finally, the experimental designs are attempts at making objects that are both visually compelling and have interesting tactile or functional features.

2.2.1 Shape Primitives and 3D Models

To act as a baseline for further testing, simple shapes were run through the pipeline. Highlighted in Figure 2.6 are a pyramid with a square base, a sphere, and a cylinder [31].

Once the technique was validated with basic shapes, the pipeline was used to experiment with more complex 3D models. Highlighted in Figure 2.7 are a chess bishop [32] and a simple cup.

2.2.2 Trace Resolution Modification

Altering the maximum resolution of the trace has an outsized effect on the thickness of the generated filaments. By halving the maximum trace size from 512 simulation units in all three directions to 256, the agents had one quarter of simulation area to move around in. However, the 3D mesh's proportions are still the same, so the end result is thicker filaments.

This modification works best on models with thin walls or features, and generally makes models more printable due to the thicker filaments. The only trade off is losing network complexity since the filaments are more dense.

As an example, this method was applied to the 3D printing benchmark, 3DBenchy [33], which is a model of a small boat with small details and thin walls. The only difference between the two results presented in Figure 2.8 is the maximum trace resolution being 512 (the default value) on the black object, and 256 on the yellow object.

2.2.3 Experimental Designs

The last set of models prepared consist of attempts to explore the limits of the objects PolyPhy could create.

The first object is a pair of torii that were arranged in slicing software such that they would print intersected and be permanently linked together once printed. Since they are held in place by each other, the effect is a pair of intersecting rings with a unique look.

The second object is a model of the famous sculpture, "The Thinker" by Auguste Rodin except half of the model has been replaced with a version of the model run through PolyPhy [29]. The effect is intended to be reminiscent of the human skeletal system.

The final object was an attempt at using Polphy to create an object that actually serves a mechanical function. A lot of household objects simply will not work for this purpose due to the hole-filled outputs - for example, a soup spoon filled with holes would not be very practical. As such, a simple box I found on Thingiverse, which is an online repository for 3D models, was used. As seen in [Figure 2.9], the box consists of a body with a lip at the top to hold the lid in place. To improve the quality of the walls, which were rendering as one layer thick, the trace resolution was decreased to improve the structural integrity. While the resulting print cannot hold liquids and does not have a lid that friction fits in place like the original model, the lid still lines up with the bottom part of the box and demonstrates a proof of concept.

2.3 Discussion and Future Work

While this technique currently has not been validated with any technical metrics, it does still present a novel and visually compelling way of creating bio-inspired scaffolding structures. Work will continue on improving the quality of these structures in addition to integrating the pipeline fully into PolyPhy as it develops, with the target of this work being the Dynamic Infill structures discussed in section 2.3.3.

2.3.1 Post-Processing and Alternative Materials

One suboptimal factor in the current mesh to printed object pipeline is the choice of fabrication device and printing material. Even though the author's personal printer was the best choice for this version of the project given available resources, filament based printing can leave behind artifacts of the process. These can range from the individual layers in the print being visible, thin strings of filament being present over horizontal gaps, to rough surfaces where the object meets support material.

A solution to this would be to commission third party printing services that have industrial quality printers to use their advanced techniques to create objects. This would invariably produce higher quality prints with the one downside being greatly increased cost.

An alternative to switching materials is to work from the perspective of getting the most out of PLA prints. The simplest way to do this is to use varying strengths of sandpaper to try and erase layer lines. However, this is not feasible given the structures present in the printed object. Another solution is to use chemicals like dicloromethane and epoxy resin to smooth the surface of the object. However, given the thin structures in these prints, the dicloromethane could dissolve integral parts of the object and epoxy resin may either get stuck in the print or not cover all surfaces.

One postprocessing method that has worked with PLA prints is to use a heat gun to blow away strings and smooth out blobs. Another method that has yet to be explored but still holds merit is using spray on varnishes as an alternative to epoxy resin to get a smooth clear finish.

2.3.2 Applications

In its current state, this project mainly has applications in science education, data physicalization, and art.

Most of the general public is unaware of the Polycephalum physarum or its algorithmic properties. The objects and pipeline in this project can be used to teach about how optimal behaviors can exist in nature and not just in a math textbook. For example, learners can measure and compare how the Monte Carlo Physarum Machine (MCPM) behaves and how objects created with it compare to those generated using conventional shortest paths algorithms.

Since PolyPhy was initially created to help astronomers visualize cosmology datasets in 3D, the pipeline presented in this thesis can be used as a physical reference or communication tool to help understand the datasets they are exploring. Holding a three dimensional object in your hands is arguably as effective of a comprehension tool as a two dimensional screen simulating three dimensional space.

Due to the inherent abstract and perceptual nature of art, there is no superior way to use this for the arts. However, the open source nature of this project allows anyone to use the tech to contribute towards any aspect of an art piece. From my own exploring, simply passing meshes through the pipeline can create visually compelling objects. For example, the chess bishop seen previously is a part of a full set that will be printed to act as a conversation piece that is also functional.

Another relevant use case for this technique is helping artists who work in the biologically inspired domain to sketch out and prototype new artworks through the use of Polyphy's generative algorithm. A theoretical future publication could involve studying how artists use the software in the creation of sculptures and other physical artworks.

2.3.3 Investigation of Structural Properties

Previously, this technique was used to create a prototype of a Physarum polycephalum inspired bicycle helmet which was presented at the ACM Symposium on Computational Fabrication in Seattle, Washington on October 27, 2022 [1]. This helmet was used to demonstrate the impact resistivity of the structures PolyPhy creates. Theoretically, PolyPhy's optimal structures should create objects that disperse impact forces evenly and effectively. This validation process presents grounds for a future publication if the mechanical properties of the object's structure can be proven.

2.3.4 Dynamic Infill

The main future technical contribution that can come out of this project is using this technique for creating a novel infill method for filament based 3D printing. While infill patterns like gyroid or rectilinear help save filament usage by minimizing the space inside of the print that needs to be filled to have a structurally sound object, they are just patterns and have no notion of the shape of the mesh itself.

Conversely, the pipeline described in this thesis is dependent on the shape of the mesh to form its structures. However, in its current state, the fits created are evenly dense at any given point. While this does work at a scaffolding strategy, it could be improved.

This improvement could take the form of a more dynamic fitting process. By having more of the central interior of the shape be sparser than the areas closest to the faces of the mesh, a supportive effect should still be achievable while lowering the volume of the support structures.

While the modifications to PolyPhy's kernel that attempt to implement this function are currently being developed, the results were not ready in time for inclusion in this document. This modification involves controlling the behavior of the simulation at the agent level, giving each one the context of where it is relative to the edge of the model and modifying its behavior appropriately. However, one model has been generated so far that indicates this technique may have some merit [Figure 2.11].

If successful, using this more dynamic version of PolyPhy could be a viable alternative to traditional 3D printing infill methods. However, this research is still in progress and falls beyond the scope of this thesis.

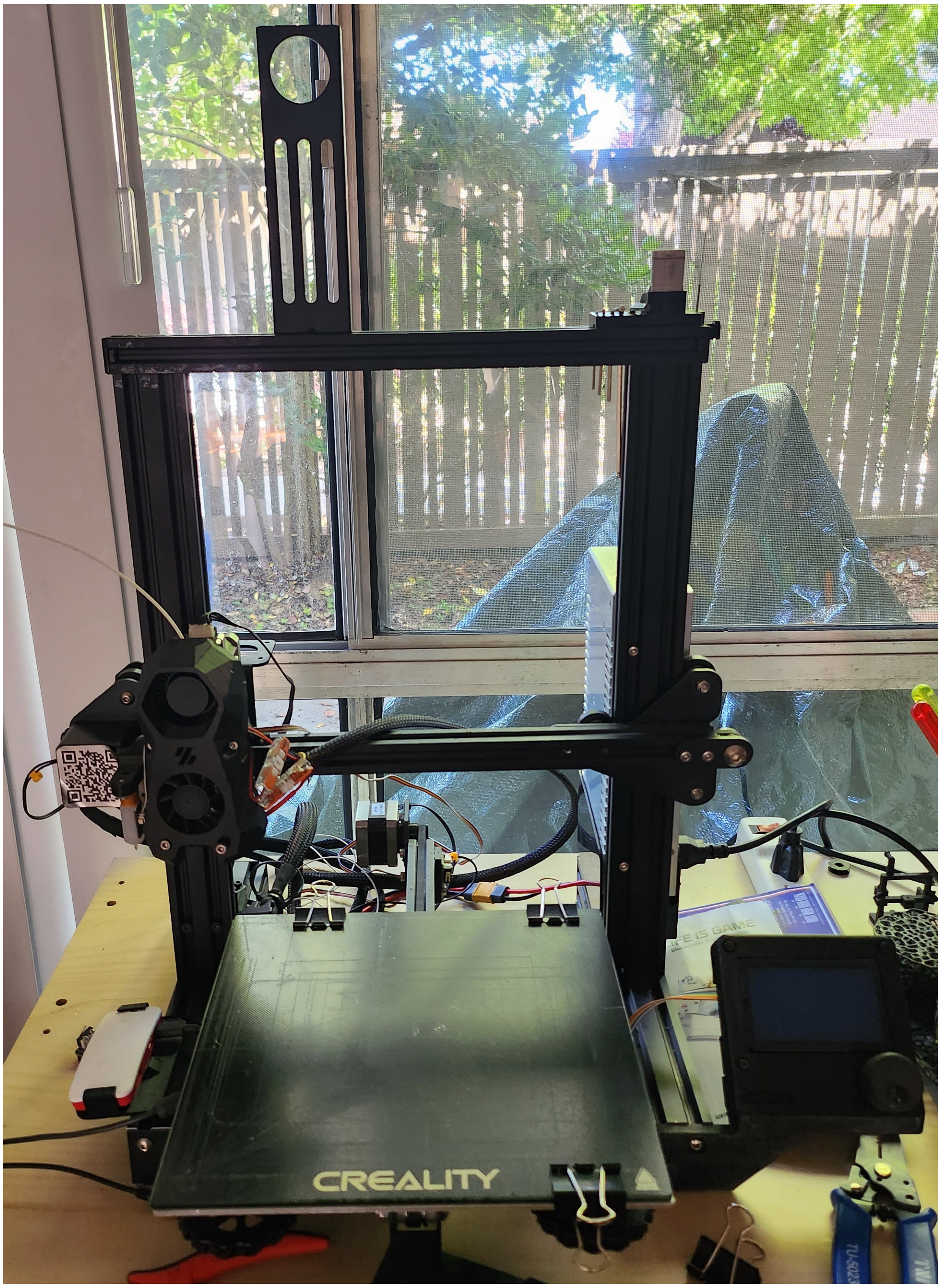

Figure 2.4: The Ender 3 filament based 3D printer used to create all objects printed for this project.

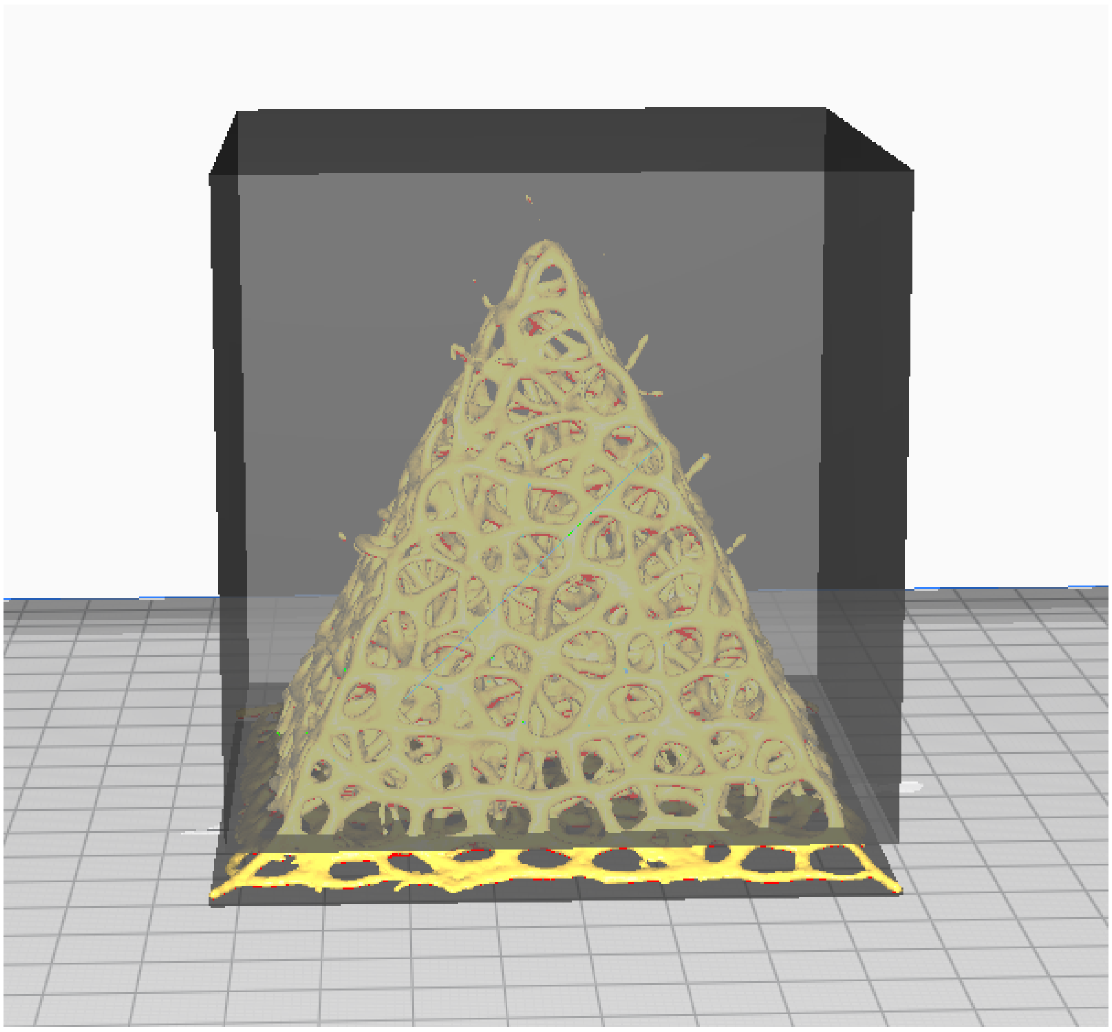

Figure 2.5: A screenshot of a support blocker being used in Ultimaker Cura to eliminate supports from generating inside the grey box as the model will print successfully without them.

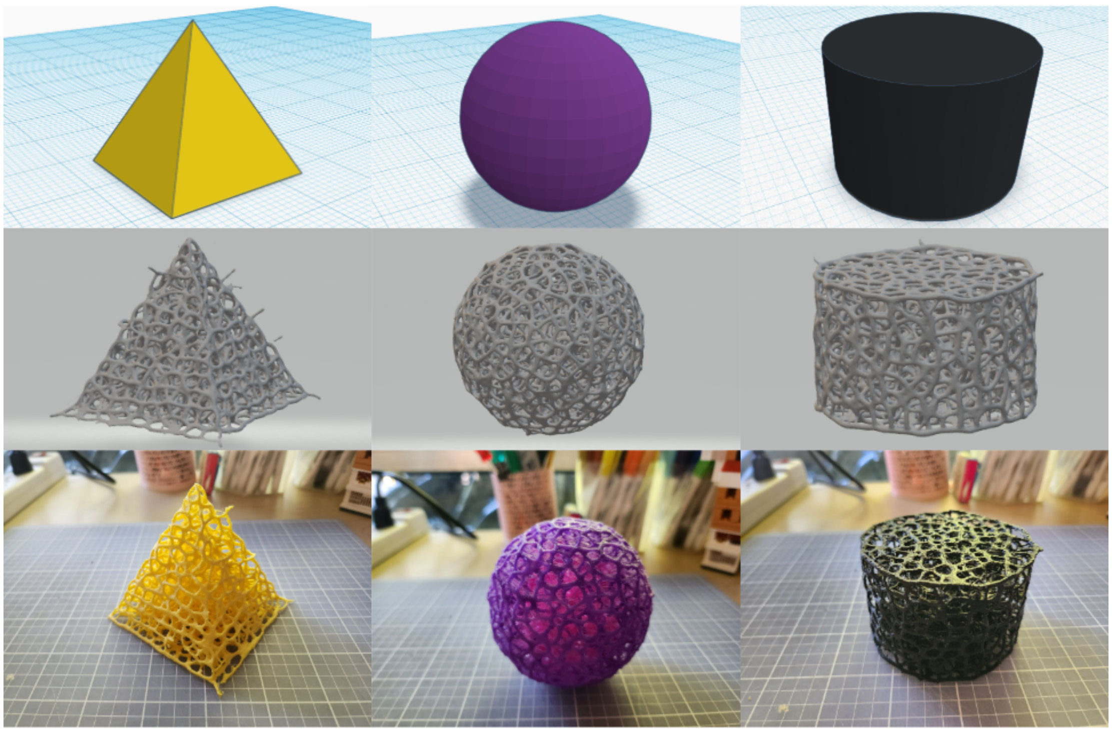

Figure 2.6: The original meshes for each shape (top row), the reconstructed mesh produced by PolyPhy (middle row), and the final printed objects (bottom row). The sphere was printed with temperature sensitive PLA that turns from purple to pink when exposed to heat. To achieve the two toned effect in the image, the sphere was heated until a full color change occured and then photographed part of the way through cooling down.

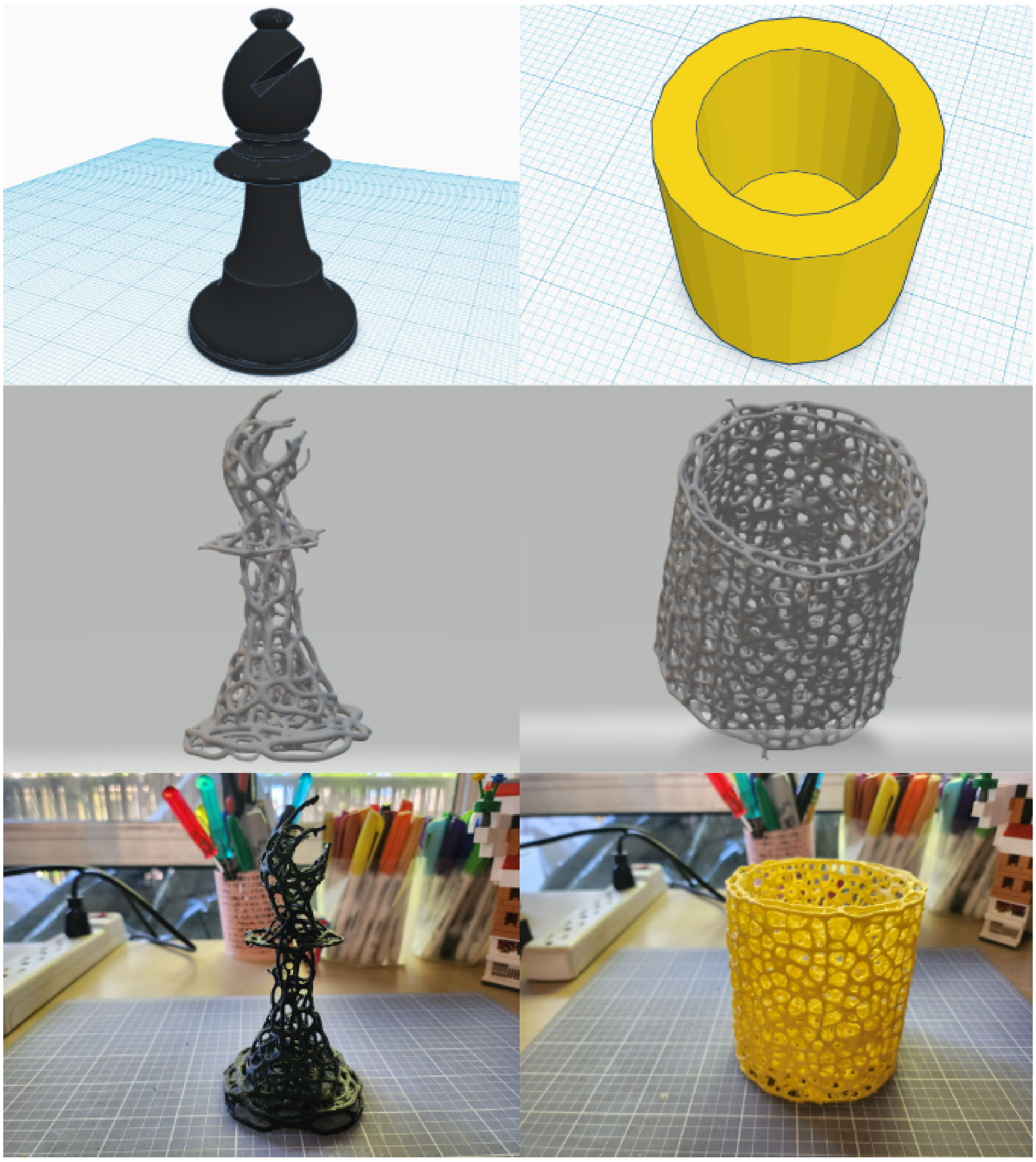

Figure 2.7: The original model, reconstructed mesh, and printed result for a chess bishop (left) and a simple cup (right).

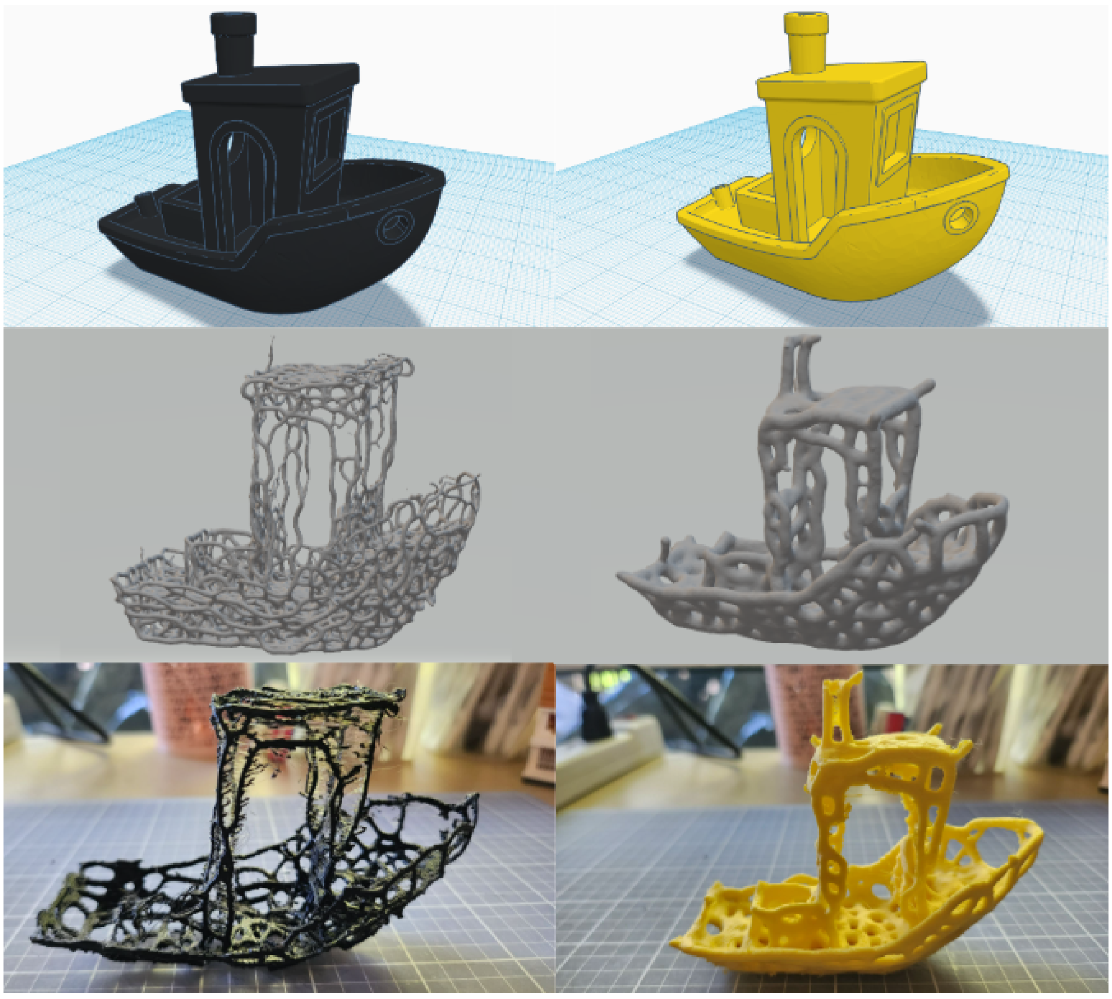

Figure 2.8: The original mesh for the 3DBench in black and yellow (top row), the reconstructed mesh produced by PolyPhy with a maximum trace resolution of 512 (middle left) and 256 (middle right), and the final printed objects (bottom row).

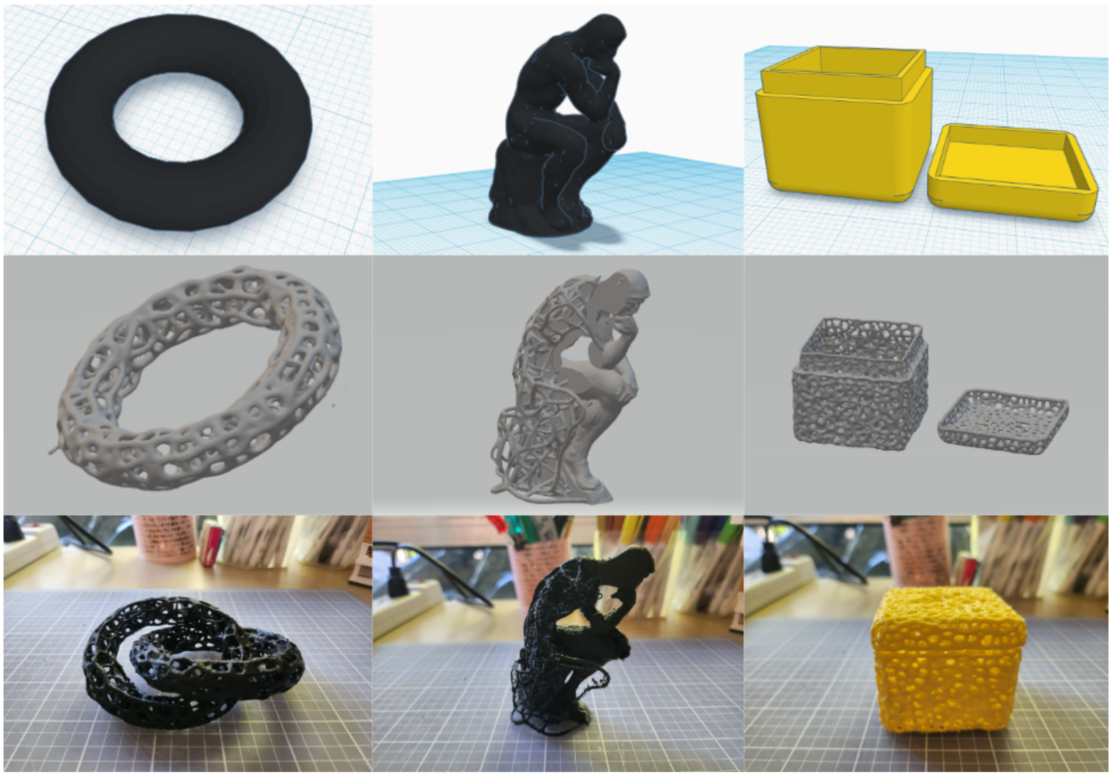

Figure 2.9: The original meshes for each shape (top row), the reconstructed mesh produced by PolyPhy (middle row), and the final printed objects (bottom row). The bottom left image is a pair of torii printed in such a way that they are linked together. The bottom middle image is of ”The Thinker” by Rodin except that half of the model has been replaced with the result of the model being passed through PolyPhy. The bottom right image is a simple box and lid.

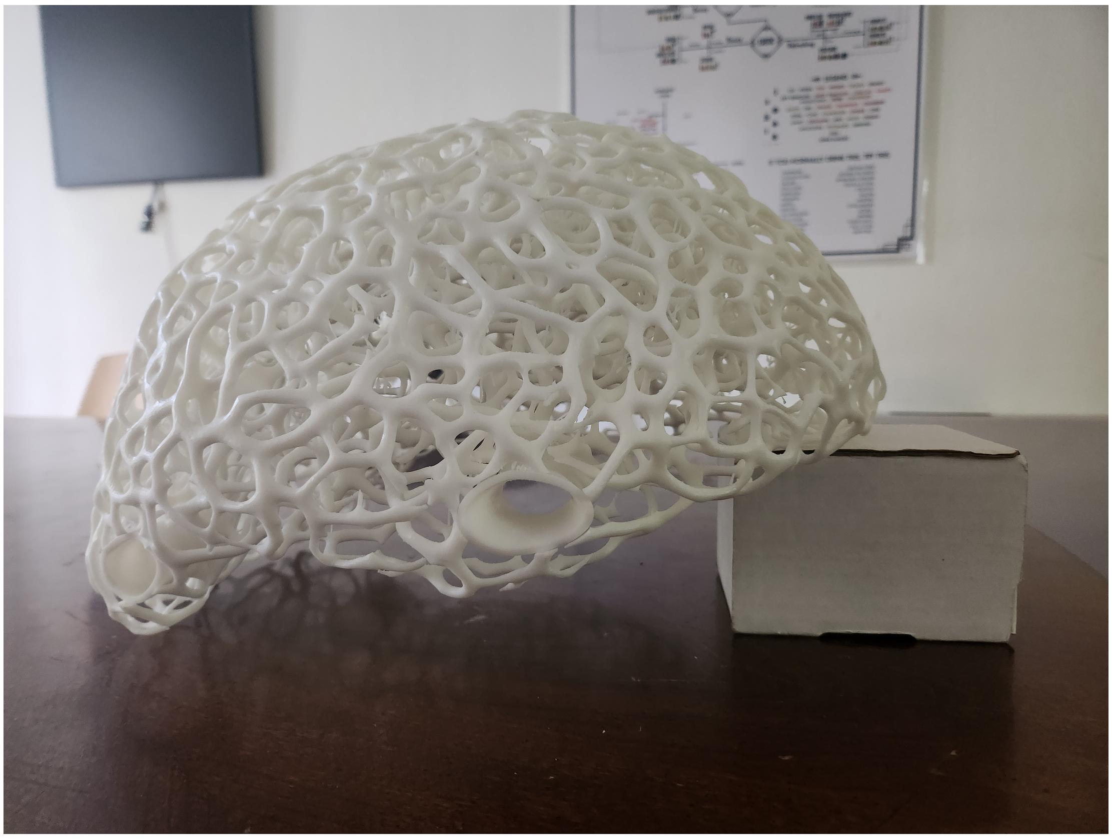

Figure 2.10: An image of the helmet presented as part of the extended abstract "Scaffolding Generation using a 3D Physarum Polycephalum Simulation" which was created with Polyphy.

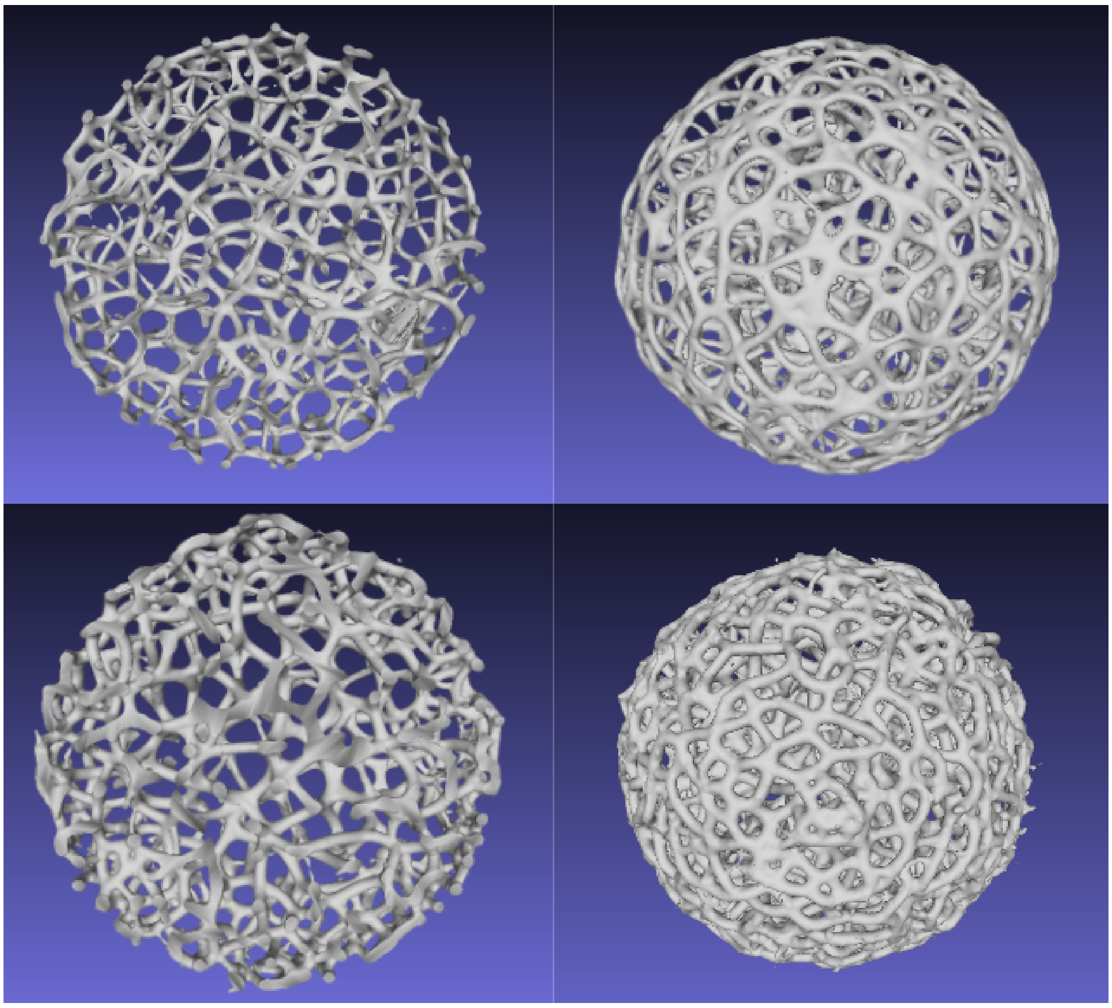

Figure 2.11: A cross section and full model of a sphere generated with the default mesh generation pipeline (top row), and a cross section and full model of a sphere generated using the experimental dynamically fitted version of the mesh generation pipeline (bottom row).

Bibliography

[1] D. Ehrlich, M. Hakimshafaei, and O. Elek, "Scaffolding Generation using a 3D Physarum Polycephalum Simulation," in *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication*, pp. 1–2, 2022.

[2] J. Høyrup, "Geometrical patterns in the pre-classical greek area. prospecting the borderland between decoration, art, and structural inquiry," *Revue d'histoire des mathématiques*, vol. 6, no. 1, pp. 5–58, 2000.

[3] Y. Abdullahi and M. R. B. Embi, "Evolution of islamic geometric patterns," *Frontiers of Architectural Research*, vol. 2, no. 2, pp. 243–251, 2013.

[4] A. Chakrabarti and L. Shu, "Biologically inspired design," *AI EDAM*, vol. 24, no. 4, p. 453–454, 2010.

[5] J. B. Rose, S. G. Natarajan, and V. T. Gopinathan, "Biomimetic flow control techniques for aerospace applications: A comprehensive review," *Reviews in Environmental Science and Bio/Technology*, vol. 20, no. 3, p. 645–677, 2021.

[6] M. P. Zari, "Biomimetic design for climate change adaptation and mitigation," *Architectural Science Review*, vol. 53, no. 2, pp. 172–183, 2010.

[7] L. Becchetti, V. Bonifaci, M. Dirnberger, A. Karrenbauer, and K. Mehlhorn, "Physarum can compute shortest paths: Convergence proofs and complexity bounds," *Automata, Languages, and Programming*, p. 472–483, 2013.

[8] V. Bonifaci, K. Mehlhorn, and G. Varma, "Physarum can compute shortest paths," *Journal of Theoretical Biology*, vol. 309, pp. 121–133, 2012.

[9] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, "Rules for biologically inspired Adaptive Network Design," *Science*, vol. 327, no. 5964, p. 439–442, 2010.

[10] T. Nakagaki, H. Yamada, and Tóth, "Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, p. 470–470, 2000.

[11] A. Adamatzky, *Advances in Physarum machines: Sensing and computing with slime mould*, vol. 21. Springer, 2016.

[12] S. Jenson, "Physarum."

[13] J. Jones, "Characteristics of pattern formation and evolution in approximations of physarum transport networks," *Artificial Life*, vol. 16, no. 2, p. 127–153, 2010.

[14] S. Simon and E. Bowman, ""the blob," a smart yet brainless organism fit for sci-fi, gets its own exhibit," Oct 2019.

[15]

[16] PolyPhyHub, "Polyphy Github Repository."

[17] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry.* Springer, 2008. 3rd Edition.

[18] J. Wu, N. Aage, R. Westermann, and O. Sigmund, "Infill Optimization for Additive Manufacturing—Approaching Bone-Like Porous Structures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 2, pp. 1127–1140, 2018.

[19] T. Tricard, F. Claux, and S. Lefebvre, "Ribbed Support Vaults for 3D Printing of Hollowed Objects," *Computer Graphics Forum*, vol. 39, no. 1, pp. 147–159, 2020.

[20] W. Commons, "Voronoi-Fry," 2020.

[21] W. Commons, "Gyroid_surface_with_Gaussian_curvature," 2006.

[22] W. Commons, "Physarum polycephalum strain lu352 - 4," 2022.

[23] O. Elek, J. N. Burchett, J. X. Prochaska, and A. G. Forbes, "Monte Carlo Physarum Machine: Characteristics of Pattern Formation in Continuous Stochastic Transport Networks," *Artificial Life*, 2021. accepted for publication.

[24] O. Elek, J. N. Burchett, J. X. Prochaska, and A. G. Forbes, "Monte Carlo Physarum Machine: Characteristics of pattern formation in continuous stochastic transport networks," *Artificial Life*, vol. 28, no. 1, pp. 22–57, 2022.

[25] O. Elek, J. N. Burchett, J. X. Prochaska, and A. G. Forbes, "Polyphorm: Structural Analysis of Cosmological Datasets via Interactive Physarum Polycephalum Visualization," 2020.

[26] R. Scopigno, P. Cignoni, N. Pietroni, M. Callieri, and M. Dellepiane, "Digital fabrication techniques for cultural heritage: A survey," *Computer Graphics Forum*, vol. 36, no. 1, pp. 6–21, 2017.

[27] W. Commons, "Hulk (40%)," 2021.

[28] D. Ehrlich, "Printing the Polyphorm: Using 3D Printing to Manufacture Biologically Inspired Rhizomatic Structures," 2021.

[29] Thingiverse.com, "Rodin's the thinker by Lampmaker."

[30] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, (New York, NY, USA), p. 163–169, Association for Computing Machinery, 1987.

[31] Thingiverse.com, "Test cylinder by SLEEPINGOWL007."

[32] Thingiverse.com, "OpenSCAD chess by timedwards."

[33] Thingiverse.com, "3DBENCHY - the jolly 3d printing torture-test by Creativetools.se by CreativeTools."