UNIVERSITY OF CALIFORNIA
SANTA CRUZ

PRINTING THE POLYPHORM: USING 3D PRINTING TO MANUFACTURE BIOLOGICALLY INSPIRED RHIZOMATIC STRUCTURES

An undergraduate thesis submitted in partial satisfaction of the requirements for the degree of

BACHELOR OF SCIENCE

in

COMPUTER SCIENCE

by

Drew Ehrlich

June 2021

The undergraduate thesis of Drew Ehrlich is approved:

Professor Angus Forbes, Chair

Professor David Harrison

Professor Jim Whitehead

Abstract

Printing the Polyphorm: Using 3D Printing to Manufacture Biologically Inspired Rhizomatic Structures

by

Drew Ehrlich

We present a pipeline for converting voxelized data generated by the Monte Carlo Physarum Machine (MCPM) based Polyphorm data visualization software into a mesh that can then be 3D printed. This reconstruction technique is based on the Marching Cubes algorithm paired with a suite of pre and post processing tools. Our process can be used to create both biomimetic structures and computational art, and can be tuned to create various visual effects based on the desired stylistic output.

Table of Contents

Acknowledgments

I would like to thank Dr. Oskar Elek for his invaluable support for this thesis and continued patience and guidance throughout this process. In addition, I'd like to thank my committee, in particular Professor Angus Forbes for helping me get involved in the Creative Coding Lab, and for allowing me to continue working in the lab through a pursuit of a master's in Computational Media starting this fall.

Chapter 1

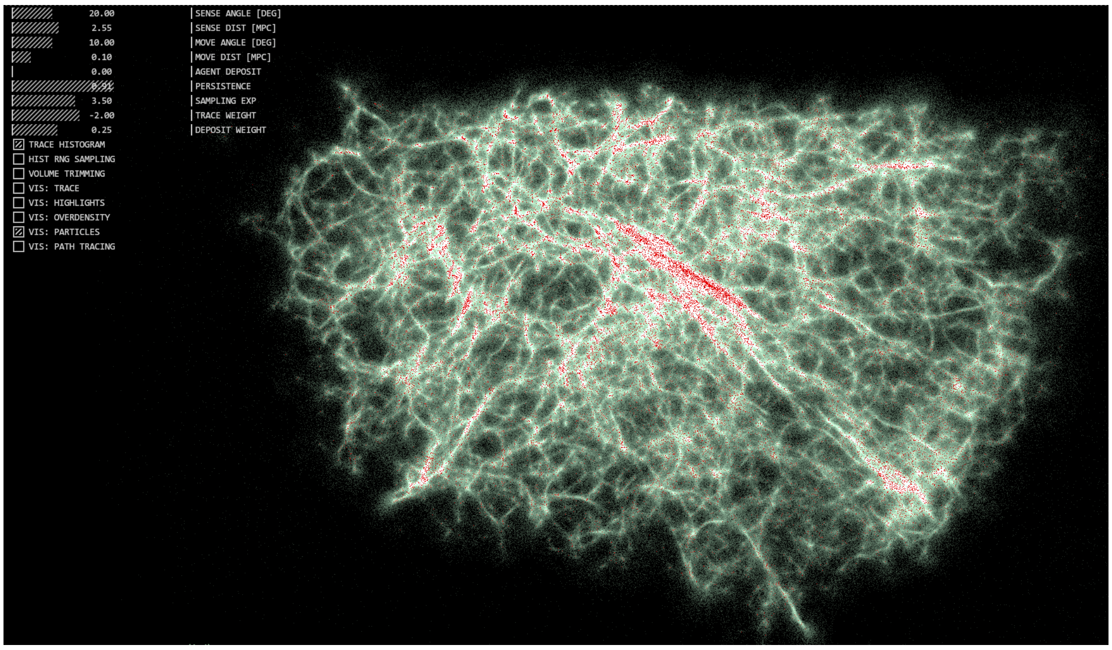
Introduction

Additive manufacturing, otherwise known as 3D printing has recently become very popular among both hobbyists and professionals alike over the past decade. Based on the idea of creating a 3D volume by extruding one very thin (usually less than .3mm) layer of melting plastic at a time, shapes are slowly created by combining hundreds or even thousands of these layers to create the desired shapes. While there are many methods designed to accomplish this task, some like selective laser sintering and polyjetting are simply out of reach for most enthusiasts simply due to the costs of the proprietary machines required to use these techniques. However, fused deposition modeling systems (otherwise known as FDM or FFF) are much more accessible and can be found on 3D printers that cost less than a midrange GPU. However, just because the cost of entry is lower doesn't mean this technique is any less effective. An extruder head that can move on three axes deposits hot material (usually a plastic) in very small and precise amounts to construct a physical shape over time.

Another important technology here is the Monte Carlo Physarum Machine (MCPM). The MCPM is an algorithm based on the behavior of the physarum polycephalum slime mold, which is a protist without cells that has been extensively documented in the past to move in an optimal manner in regards to acquiring resources. The MCPM takes this behavior and applies it 3D point sets, generating visually intruiging computational visualizations in Polyphorm.

Initially designed with cosmological research in mind, Polyphorm can use any well formatted dataset to create compelling transport networks that can serve as both art and act as optimal structures. Even though these kinds of designs were not its original purpose, Polyphorm's structures are still applicable to this case since the MCPM still allows it to create optimal transport networks regardless of the source of the input data. These transport networks represent how data travels through the structure, and since these data are looking to minimize their traveling time from place to place, these networks are optimal - in a way resembling a shortest paths solution in three dimensions.

This software paired with 3D printing inspired two distinct tasks - the first being finding a way to create a 3D printed representation of the simulations in Polyphorm, and the second being to explore the possibilities of using Polyphorm to create biomimetic (meaning mimicking biological structures or processes) structures that could be used to help minimize material use in 3D prints. However, not all shapes that Polyphorm generates are printable - an important part of this thesis is to determine the best ways to tune the simulation controls to create printable shapes that have strong and clear connections between them.



```
20.00    SENSE ANGLE [DEG]
2.55     SENSE DIST [MPC]
10.00    MOVE ANGLE [DEG]
0.10     MOVE DIST [MPC]
0.00     AGENT DEPOSIT
0.91     PERSISTENCE
3.50     SAMPLING EXP
-2.00    TRACE WEIGHT
0.25     DEPOSIT WEIGHT
```

☑ TRACE HISTOGRAM
☐ HIST RNG SAMPLING
☐ VOLUME TRIMMING
☐ VIS: TRACE
☐ VIS: HIGHLIGHTS
☐ VIS: OVERDENSITY
☑ VIS: PARTICLES
☐ VIS: PATH TRACING

Figure 1.1: A screenshot taken of Polyphorm

1.1 Goals

The goals for this thesis are as follows:

- Tune Polyphorm's simulation to represent a printable structure
- Take the data that Polyphorm can export and reconstruct it as a 3D mesh
- Take this mesh and 3D print it
- See what kind of shapes could be generated using this method, and what kind of structural benefits the reconstruction process might have

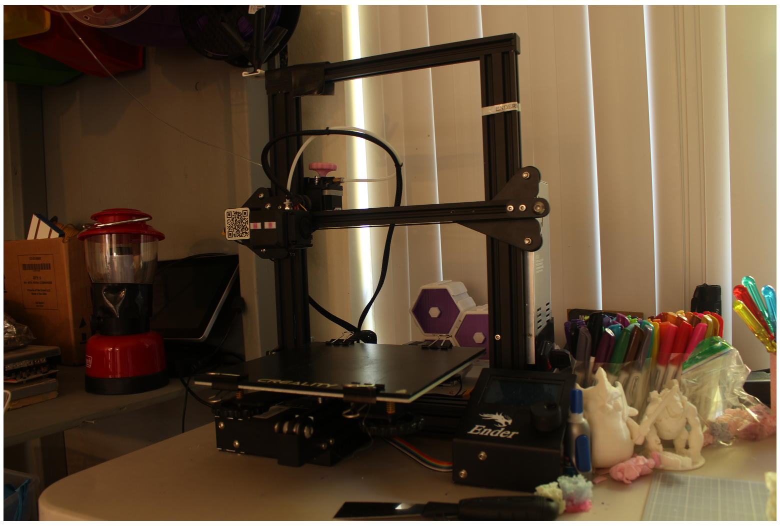


Figure 1.2: The Ender 3 printer

Chapter 2

Background

2.1 Defining Key Terms

Before going any further, our pipeline uses many not commonly known techniques and tools. This section will help clear up these terms for more clarity in analysis of our technique.

2.1.1 Monte Carlo Physarum Machine

The Monte Carlo Physarum Machine (MCPM), is the algorithm that backs Polyphorm and determines how points in density fields should be connected. The algorithm is built to model the behavior of the Physarum polycephalum 'slime mold', and should allow for optimal connections of points due to the nature of the Physarum polycephalum needing to be as efficient with resource gathering as possible to survive.

2.1.2 Polyphorm

Polyphorm, created by Oskar Elek in the Creative Coding Lab at UC Santa Cruz is a piece of software that displays connections in a density field which are determined by the MCPM algorithm. [1] Initially intended for use with cosmological data, it can work on any set of well formatted data, and creates unique visual networks that theoretically have optimal substructure.

2.1.3 Density Field

A density field is a way to use probability to describe the chances of finding a piece of data at any specific point, with the areas with increased density representing increased probability of data being present at any given time. In terms of Polyphorm, the generated simulation from the MCPM is a density field represented in 3D space with the denser areas of a shape representing the increased odds that agents will be found in that space at any given time.

2.1.4 Gaussian Filter

Used here for reducing noise when creating a final mesh, a Gaussian Filter is an algorithm that creates a visual blur, and in our case just seeks to remove unnecessary noise. Any photo editing software that has a blur filter is almost certainly based off of this, and in this case it is simply applied in 3D. [2]

2.1.5 Marching Cubes Reconstruction Algorithm

The Marching Cubes Algorithm is a method of reconstructing a mesh from a set of points that functions on a very specific set of rules. The algorithm iterates through every possible voxel in the object, and for each voxel determines which corners of the voxel are 'in' the shape and then draws triangles inside the voxel based off of which specific corners are marked as part of the final shape. The triangles added are predefined, coming from a set of 15 corner combinations.[3]

2.1.6 Slicer

A slicer is a piece of software that prepares 3D models for 3D printing by turning their shape into a series of mechanical movements the printer can understand and execute. Slicers are also used to define the settings the printer should use while printing, like print speed and how thick each layer of filament should be.

2.2 Related Work

Over the past decade there has been relevant research to this topic in regards to creating more optimal infill patterns for 3D printed models, which is something our technique hopes to address due to the MCPM creating optimal structures.

The broadest of these papers is 'OpenFab: A Programmable Pipeline for Multi-Material Fabrication' by Vidimče et al. (2013) which creates a framework for using many kinds materials and structures inside of 3D prints for both artistic and structural effects. [4]

Another relevant paper to this topic is 'Global Stiffness Structural Optimization for 3D Printing Under Unknown Loads' by Wang et al. (2016). This work lays out a new method for creating optimal infill patterns that minimize material usage to create more efficient 3D prints. [5]

In a similar vein to Wang et al., 'Density-based topology optimization for 3D-printable building structures' by Vantyghem et al. (2018) has a similar goal of creating optimized structures except in this case for use in construction and architecture. The authors here more focus on topology and the properties of the materials they use that the shape of the structures themselves. [6]

The most relevant of these is 'The Fabric of the Universe: Exploring the Cosmic Web in 3D Prints and Woven Textiles' by Diemer and Facio. The paper also dicusses taking 3D density fields and converting them into printable structures, but they generate the data in a different way and their main application is textiles. [7]

While all of these works cover more ground than this one does, they all serve as good jumping off points for learning more about infill optimization and optimal structures. However, we are still covering new ground since none of these processes have been applied to the MCPM and Polyphorm or in the context of an algorithm built off of the behavior of the physarum slime mold.

Chapter 3

Methods

3.1 Preprocessing for Polyphorm

There are two ways to go about selecting a dataset to give to Polyphorm. The first is to take any OBJ and run it through a script bundled with Polyphorm that turns it into a set of points that the software can understand. The other option is to use one of the datasets built into Polyphorm that are based on a variety of distributions or shapes. These include mathematical distributions like a Poisson Distribution, and cosmological ones like the datasets from the Sloan Digital Sky Survey or the Bolshoi-Planck simulation. While some OBJ files may perform better than others, anything with vertices should be compatible with the software, since those are the values that are taken from the OBJ file to prepare it for use with Polyphorm.

Another important consideration is the number of vertices in the model being used. Depending on how sparse the resulting mesh is desired to be, we found it appropriate to either decimate or subdivide the vertices of the OBJ file, either lowering or raising the vertex count respectively. The results from a point set with too vertices will result in a vastly different look from a point set with too many vertices, even if they are tuned the same way inside Polyphorm.

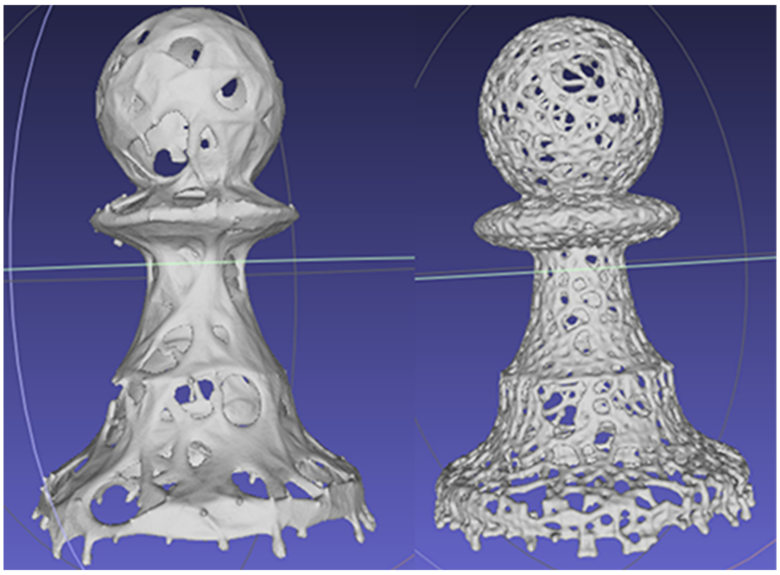


Figure 3.1: Resulting meshes from chess pawns with the original amount vertices (left) and double the amount of vertices (right)

3.2 Tuning Polyphorm's Visualization of the model

Once the data has been loaded into Polyphorm, the visualization can be tuned. The goal of this process is for there to be visible connections between all the points so the final model can have a solid structure and integrity. To get these visible connections, there are four major variables (controlled by sliders) that can aid in the process of tuning the simulation. These variables are sense angle, sense distance, move angle, and move distance.

Since Polyphorm is based on an algorithm that creates many independent agents, these sliders govern their behavior and change the way they move around the simulation. Sense angle and distance govern what each agent can 'see' in the process of looking for other points to move towards. Sense angle defines how wide this range is in degrees, while sense distance defines how far away points can be detected. With sense angle and distance covering what the agent can detect, move angle and distance define how the agents can move towards other points. Move angle defines the an area of points the agent can move to, which move distance defines how far away points in this range can be to travel to. While they may seem similar, the distinction between these sensing and moving is that the points the agent can move to is usually a subset of the points the agent can detect. The reason that both metrics exist is that agents can move a very long distance over the course of the simulation, and it's important for them to be aware of more than just the points they can move to help optimize the paths they create through moving through the density field.

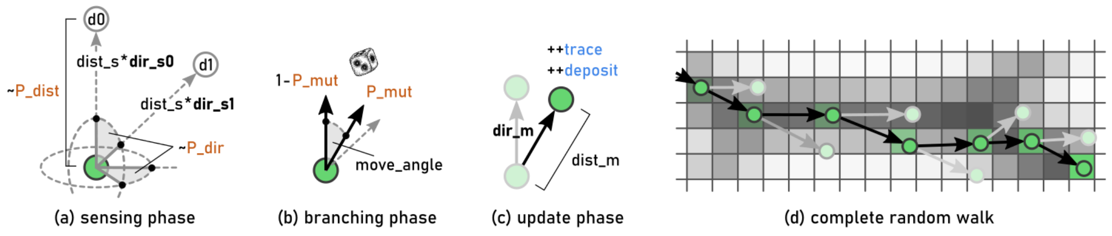


Figure 3.2: How agents in the MCPM move and how move angle, move distance, sense angle, and sense distance are involved. Reprinted with permission from [8].

While there are some other values that can be changed in the simulation, these can usually be left alone after being set to a specific default value. These values are persistence and agent deposit, which both control different aspects of how paths between points are visualized.

Most tuning can be done by just looking at the simulation and adjusting sliders until the desired look is achieved, and there are some good rules of thumb for this process. These being that sense angle should be around double the value of move angle, move distance should be in the .03-.05 range, persistence should be in the .91-.94 range, and agent deposit should usually be set to zero. For most inputs, these setting should create results that get you past a basic tuning.

3.3 Exporting the Data for Reconstruction

In Polyphorm, you can press F5 or F6 to pause the simulation and export the current state of the agents as a list of voxels. This creates a binary file that is used to generate the mesh later on, and a metadata file that includes useful information, like the size of the point cloud in voxels. This is the only format Polyphorm can export voxels too, making this pipeline necessary since this proprietary format needs to be translated into something a 3D printer can understand.

3.4 Creating a 3D Mesh from the Binary Export Data

The goal of this process is to take the exported voxels from Polyphorm and use them to create a mesh that can be 3D printed.

3.4.1 Reshaping the Data

The first step in this pipeline is to take the binary file that Polyphorm provides and read the raw data into a three dimensional numpy array. This array is then reshaped into a specific three dimensional shape as defined by the metadata file that was created alongside the binary file. This is important since if the dimensions used do not match those in the metadata, then the mesh won't be created properly as the voxels won't be spaced out as intended, since the dimensions of the shape will be swapped and the result won't look as intended.

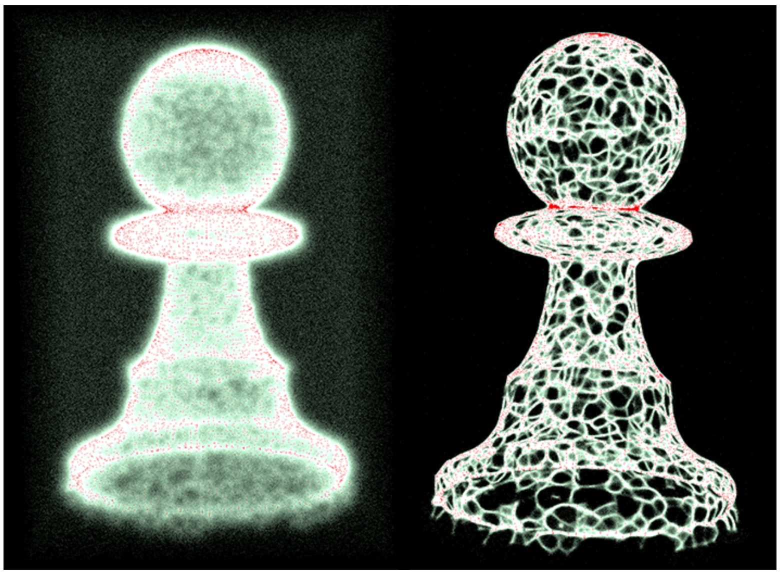


Figure 3.3: The chess pawn dataset in Polyphorm before and after tuning the simulation

3.4.2 Using Marching Cubes to Create a Mesh

Using the PyMCubes Python library, we were able to pass in our reshaped numpy array to a function and get a 3D mesh from it. [9] However, there were a couple additional steps taken to help increase the quality of the resulting meshes. The first is to apply a Gaussian Filter to the data to reduce existing noise in the data, and the second was to experiment with the threshold that takes the Marching Cubes function takes in. This threshold determines what data is 'inside' the shape, and what is 'outside' based the densities of the points. For this specific function, the lower the threshold the less points included in the shape, and vice versa.

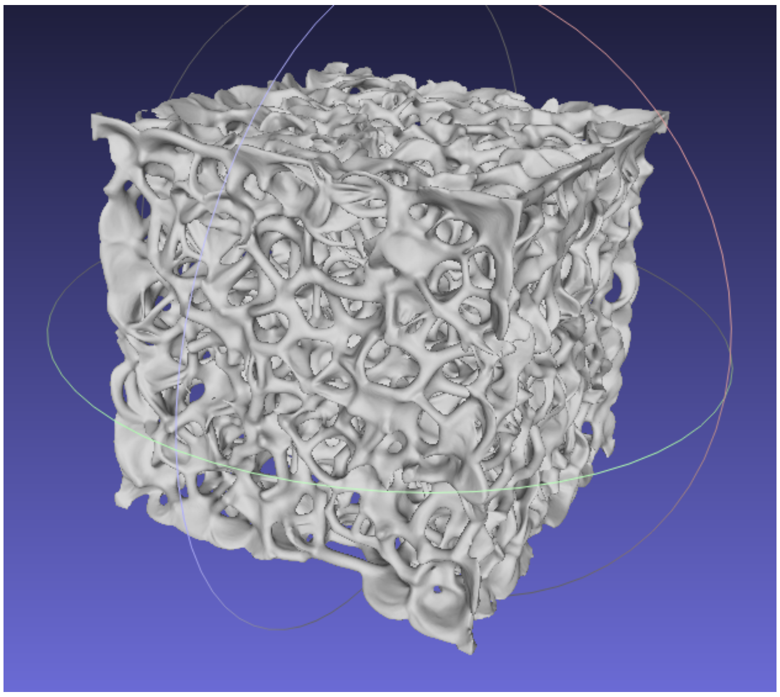


Figure 3.4: A shape created by the Marching Cubes algorithm generated from a Bolshoi-Planck simulation

Chapter 4

Results

4.0.1 Optimizing the Gaussian Filter and Marching Cubes

When it came to deciding the parameters to use for blurring and thresholding the Marching Cubes function, determined that in most cases a blur of (1.2, 1.2, 1.2) was the best general purpose value. However, for some models, extra tuning was necessary based on the density and amount of vertices in the shape. In addition, we determined that the best value to use as a threshold for the marching cubes function was 17, as it balanced removing extra noise from most shape and not cutting into material that belonged to the shape itself. Like tuning for the blur, some models benefitted from different thresholds for similar reasons, but values beyond 17 give minimal discernible difference, and values smaller than it tend to eat away at the shape at an accelerated rate.

4.1 The Printing Process

There are two main steps in turning a 3D model into a 3D print, slicing and using the printer itself. Both step aren't necessarily straightforward and do require some attention and tuning to get right.

4.1.1 Slicing the Reconstructed Model in PrusaSlicer

Where there are many slicing softwares, the software we chose to use was PrusaSlicer for previous experience and ease of use, but in general any slicer that can

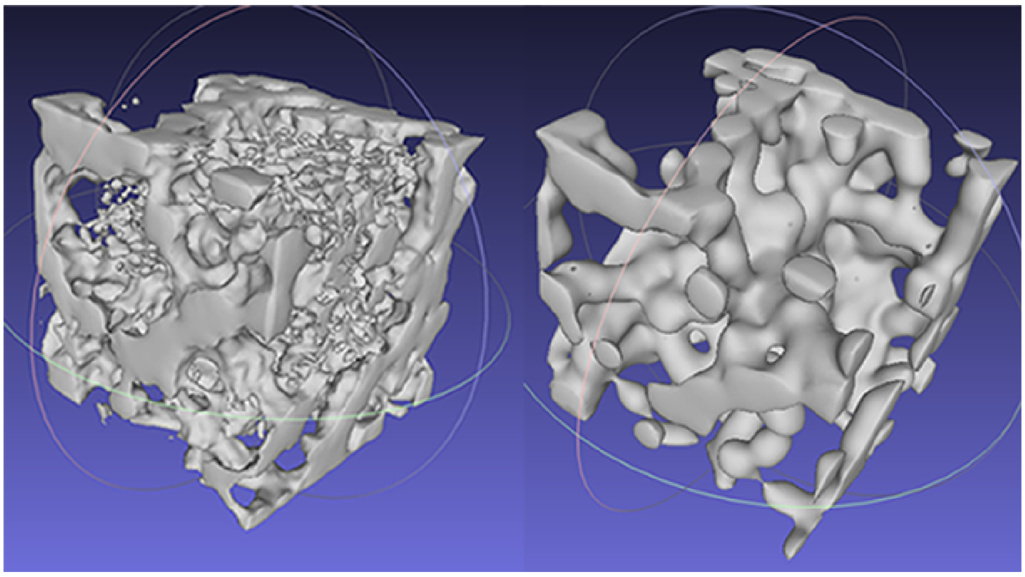
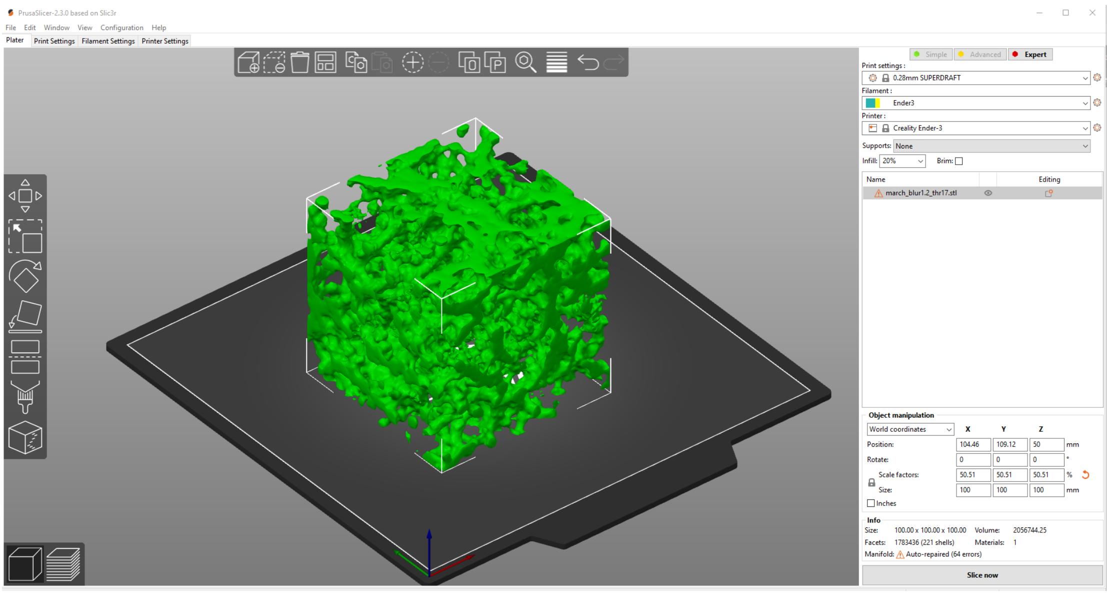


Figure 4.1: Cubes based on the same data generated using thresholds of 10 (left) and 20 (right)

support your printer will work for this pipeline. The settings we used were built off of PrusaSlicer's presets for the printer we used, the Creality Ender 3.

To even begin using the slicer, the intended mesh must be watertight and not have incomplete geometry. Since our technique usually resolves this (and if it cannot, the automatic repair tool inside of PrusaSlicer does a decent job of patching up open geometry), we can move forwards with the slicing process. The main settings that we considered were layer height, infill, supports, and use of brim. Layer height is up to the user, however we used .28mm layer heights for speed of printing but lower layer heights will lead to more precise results. We chose to use no infill or supports since the shapes produced by our pipeline usually don't have large angles or overhangs and should be able to stand on their own. The lack of supports are also inspired by the fact that these shapes should also have optimal structure due to the nature of the MCPM algorithm being based off of a slime mold that moves optimally. In cases where support is necessary due to the model having large overhangs, PrusaSlicer has a 'paint-on supports' tool that allowed us to only put supports on the overhanging area since the rest of the model should not need them.

We also scaled the models down by large amounts to reduce printing time, however most models should be able to printed at any size. However, models generated using shapes with large overhangs may need supports at a certain scale even if they can print at smaller sizes.



Print settings : 0.28mm SUPERDRAFT
Filament : Ender3
Printer : Creality Ender-3
Supports: None
Infill: 20%
Brim: ☐

Name	Editing
march_blur1.2_thr17.stl	

Object manipulation

World coordinates	X	Y	Z	
Position:	104.46	109.12	50	mm
Rotate:	0	0	0	°
Scale factors:	50.51	50.51	50.51	%
Size:	100	100	100	mm

☐ Inches

Info
Size: 100.00 x 100.00 x 100.00 Volume: 2056744.25
Facets: 1783436 (221 shells) Materials: 1
Manifold: Auto-repaired (64 errors)

Figure 4.2: A screenshot of PrusaSlicer

4.1.2 Printing the Final Result

The final step in our process is to print the model itself. This lies mostly in the hands of the end user, but external factors like the height of the printing surface not being calibrated properly or the print not adhering to the bed can cause failed attempts. In our testing, we used PLA on a Creality Ender 3, which uses fused deposition modeling, to print our models with extruder temperatures ranging from 205 to 210 degrees celsius and bed temperatures ranging from 55 to 60 degrees celsius. This technique should still work on more advanced machines like those that use SLS or polyjetting, we simply did not have access to these kinds of resources for this testing.

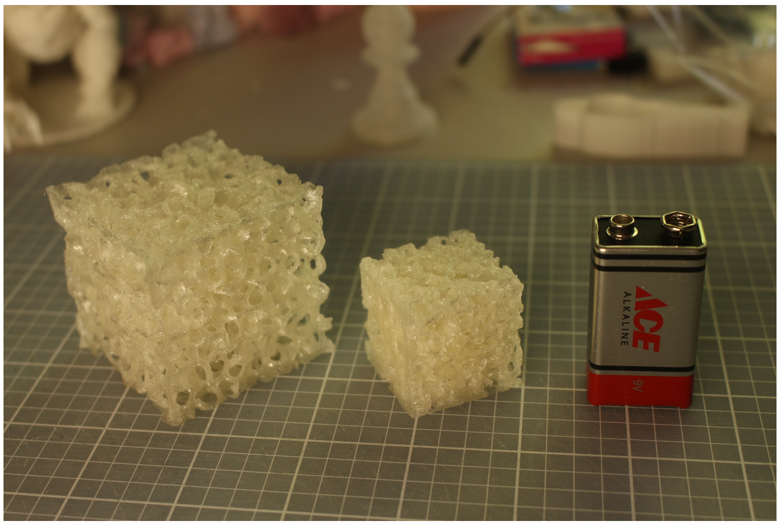


Figure 4.3: A comparison of sparse and dense versions of Bolshoi-Planck simulation data printed in clear PLA with .24mm layer height and no supports. 9 volt battery for scale.

Chapter 5

Discussion

5.1 What Kinds of Shapes Worked Best?

In our testing we noticed that shapes that didn't have any large flat surfaces were treated the best by the algorithm. For example, a 3D model of Rodin's 'The Thinker' we used didn't have any large flat surfaces (save for the bottom which was treating as an opening), and printed easily. However, we had issues on the 3D printing benchmarking print Benchy. The issue with Benchy was that the deck of the boat wasn't really recognized as having points by Polyphorm and these point were therefore not included in the mesh leading to the resulting shape being neither accurate nor printable.

However, unlike flatter surfaces, very irregular shapes and structures work very well. For example, a model based off of a sculpture called 'Zoe' by Colleen Flanigan and a model based off of the SDSS cosmological dataset both led to excellent reconstructions that were both printable at high levels of detail and represented the original datasets well even at lower resolutions.

For most of our initial testing, we used subsections of a large cube generated using a Poisson distribution in three dimensions. While this also worked very well with this method, an interesting artifact from our pipeline was areas where connections were chopped off by the selected subsection ending were left open, which disrupted the printability and structure of the model. However, this could be resolved by setting the values along the sides of the cube to be zero, which meant that they were ignored by the algorithm, however the model no longer had open sides.

5.2 Possible Further Applications of this Pipeline

There are two possible further applications for this pipeline that we're in the process of pursuing. The first is creating computational art by using this process on existing ideas and concepts and giving them a more biological inspired look. For example, one of the models we created was based from a chess pawn along with the previously mentioned 'The Thinker' sculpture.

The second area we're considering is using the generated structures to mimic biomimetic structures for use as potential new infill methods for 3D printing. This would be significant since a lot of work has been done in the area of replicating nature's techniques for creating optimal structures (such as the behavior of the physarum slime mold), and there are very promising possibilities that it will be useful for creating new infill patterns and methods for 3D printing.

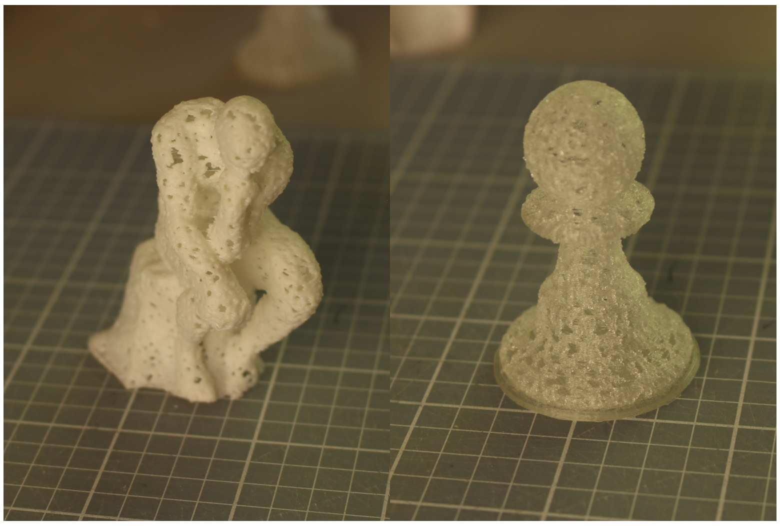


Figure 5.1: Rodin's 'The Thinker', white PLA at .24mm layer height and no supports (left). A chess pawn printed to scale, clear PLA at .24mm layer height and no supports (right)

5.3 Failed Initial Methods

An initial attempt at creating our pipeline involved using a technique called Screened Poisson Reconstruction, and required a lot more hoops to jump through including generating normals for vertices and manually creating the OBJ file in our script. While this method worked, the results weren't as strong as they could be, and the switch to using marching cubes led to strictly better results. Marching cubes allowed for much more thinner and detailed filaments in the resulting shapes that screened Poisson reconstruction simply ignored. While screen Poisson reconstruction still creates shapes that could serve as computational art, it misses the mark in terms of detail and resolution we are seeking in our search for recreating biomimetic structures.

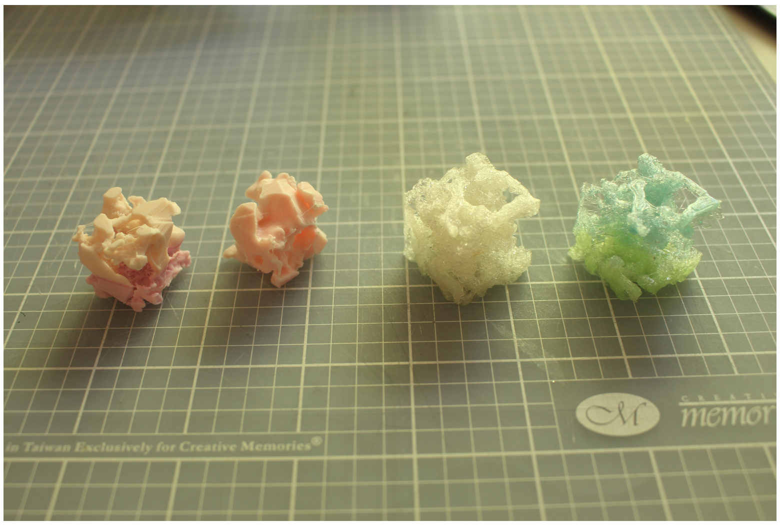


Figure 5.2: Less optimal results from screened Poisson reconstruction (left) versus Marching Cubes (right). Various colors, PLA, .16-.24 layer heights, no supports.

5.4 Areas to Improve the Pipeline

While most of the pipeline is somewhat optimized at this point since the only computationally intense task, the marching cubes algorithm itself, uses a function from an open source library. There are two distinct areas that could possibly be improved. The first is this marching cubes function, and the improvement would involve rewriting the algorithm from scratch, focusing on changing the behavior around irregular shapes to better suit the kinds of models we used.

The other place to optimize is in Polyphorm itself, which is still being developed and will be receiving support from the UCSC Center for Research in Open Source Software this fall. There are currently advances in de-noising the simulation, but these updates are not publicly available yet, and are still being tested. Another change that could be made to Polyphorm would be to modify the MCPM to specifically not allow for the rendering of overhangs that won't be printable without support in an effort to increase the amount of meshes that will be printable without need for supports.

While more advanced printing technologies could also help yield better results, we simply do not have access to machines that can perform anything more advanced than FDM printing. This is an area that can be explored in future work if that status quo happens to change, as SLS and polyjetting are both very promising technologies.

Chapter 6

Conclusion

We have a created a pipeline to take the exported data from Polyphorm and turn it into a 3D printable mesh. These models can serve both artistic and structural purposes, as the MCPM generates optimal structures itself. This work is significant because Polyphorm previously did not have a way to convert its output into a 3D mesh, and the results show promise for creating new methods for making optimal internal structures for 3D prints.

6.1 Future Work

In the near future, we are planning to make a more focused investigation on the prospects of using this technique to propose new infill patterns and methods to help optimize various kinds of 3D structures for various factors including minimizing material and maximizing structural integrity.

Bibliography

[1] O. Elek, J. N. Burchett, J. X. Prochaska, and A. G. Forbes, "Polyphorm: Structural analysis of cosmological datasets via interactive physarum polycephalum visualization," 2020.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006.

[3] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, p. 163–169, Aug. 1987.

[4] K. Vidimče, S.-P. Wang, J. Ragan-Kelley, and W. Matusik, "Openfab: A programmable pipeline for multimaterial fabrication," *Commun. ACM*, vol. 62, p. 97–105, Aug. 2019.

[5] T. Y. Wang, Y. Liu, X. Liu, Z. Yang, D. Yan, and L. Liu, "Global stiffness structural optimization for 3d printing under unknown loads," *Journal of Computer Graphics Techniques (JCGT)*, vol. 5, pp. 18–38, August 2016.

[6] D. C. W. S. M. e. a. Vantyghem, G., "Density-based topology optimization for 3d-printable building structures," *Structural and Multidisciplinary Optimization*, 2019.

[7] B. Diemer and I. Facio, "The fabric of the universe: Exploring the cosmic web in 3d prints and woven textiles," *Publications of the Astronomical Society of the Pacific*, vol. 129, p. 058013, apr 2017.

[8] 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.

[9] pmneila, "Pymcubes." https://github.com/pmneila/PyMCubes.