# Conical Slicing

## A Mathematical Solution to 3D Printing's Biggest Problem

Ezra Huska, Grade 9
Fort Richmond Collegiate
99 Killarney Ave, Winnipeg, MB, R3T 3B3

## Abstract

3D printing has had the problem of printing overhangs since its very day of conception. This has put limits on what 3D printers can print. This project explores a mathematical way of overcoming this problem called Conical Slicing. Conical Slicing differs from regular slicing in that regular slicing makes layers, and Conical Slicing makes shells. The idea is that, in the same way that you would build up a part layer by layer, you do it instead in these angled shells. This allows you to print practically in mid-air but still have it supported from the sides. This concept was originally developed by some students at the University of Applied Sciences in Winterthur, Switzerland, but it came with many problems. Through this project, I have made the concept come to fruition and made it accessible to millions of 3D printer users. I have also made the code significantly faster in some places, which has resulted in a 2000% increase in the speed of the code. This project shows the promise of this technology and its great effect on 3D printing as a whole, along with the hundreds if not thousands of industries that use 3D printing to manufacture some of their products, allowing them to print more advanced parts.

## Table of contents:

# 1. Introduction

Manufacturing has come a long way from the past, where you had to make metal moulds for every single part you wanted to make, to being able to make almost anything in your garage using a 3D printer. But, 3D printing has long since had the issue of printing overhangs. In short, overhangs are where the printer tries to print with nothing below it. This leads to catastrophic print failure. For over 40 years, the common solution has been to use significant extra plastic to create a lattice structure underneath the overhang so the printer has something to print on. This leads to significant plastic waste, for some parts, over 40% of the plastic is thrown away as support material.

The problem of overhangs not only leads to pounds of plastic being wasted every year but it also reduces the usefulness of 3D printing. Even with the use of support material, overhangs end up with ruff undersurface and unflattering parts. This comes as a detriment to the product engineering industries as they often use 3D printing for rapid prototyping. Ruff undersurfaces, and or reduced use of overhangs reduce the quality and usefulness of the parts created by them.

In this project, I will explore an innovative solution to these problems in the form of Conical Slicing and some of the downsides that still must be worked out.

# 2. Background Information

To understand this project better, you must know some common 3D printing vocabulary.

**STL** - STL files store 3D models in the form of triangles, with 3 points on each. With this, you are able to accurately store 3D information in a computer

**G-code** - G-code is a file format that carries moves for a 3D printer to do. This is usually done in one of two ways, either as a series of points to go to, this way often takes up more space but is more accurate. The second way is to think of g-code points as the beginnings and ends of lines, this way uses less storage space but is significantly harder to do anything with afterwards.

**3D Slicer** - A program that takes a 3D model, usually in the form of an STL, and splits it into hundreds of slices that a printer can print (Figure 1). Then, it packages them up into a G-code file.

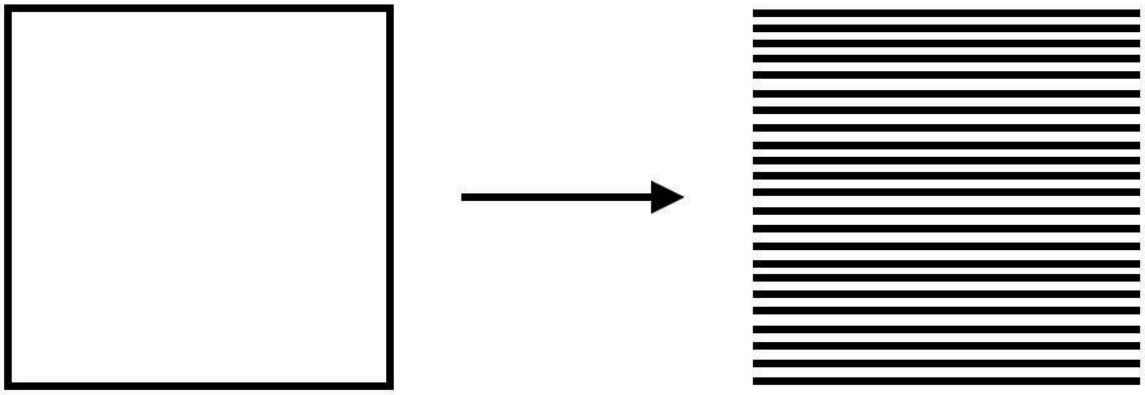

Figure 1:
A depiction of
modern 3D Slicers

**Overhang** - As I alluded to in my introduction, an overhang is basically when a part of the model has nothing under it for support (Figure 2), and due to the way 3D slicers currently work, the printer ends up printing mid-air and ruining parts.

**Support Material** - The current solution to the problem of overhangs consists of building up plastic below it layer by layer, so you end up not printing in the air (Figure 3).

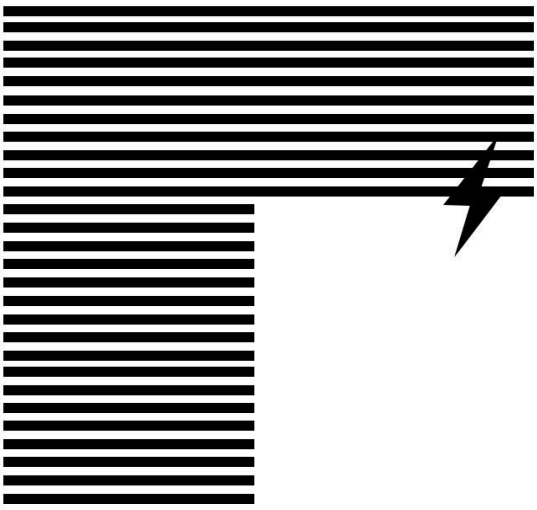

Figure 2
A depiction of an overhang

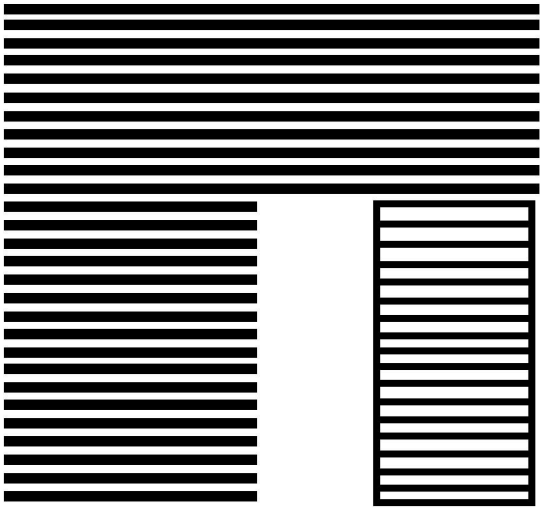

Figure 3
A depiction of an overhang with a
support structure

# 3. Existing Prototypes

Now I must say that this concept is not a completely new idea; it was attempted by the University of Applied Science in Winterthur, Switzerland in 2021. They made a promising attempt, and the idea was picked up by CNC Kitchen, a famous 3D printing-related YouTube channel. He used their code and even made some examples to print. But then the project went dry, and nobody really tried to go further.

I am almost certain that their project failed to take off because of the many that it has, some of which include:

- **Slicer Constraints:** The code only worked with a paid slicer called Super Slicer. It is one of the least common slicers. Almost no one uses it unless they are a student. Not using a common slicer was one of the main downsides to their project.
- **Complexities:** They convoluted the code and mixed it with another project making it more complicated and less functional. This made a bad first impression of Conical Slicing. Not only was the code complicated there was very little documentation. The factors together dissuaded many people from even trying to use what they had created, and the fact that you had to do so much work to get it working turned many people away.
- **Inefficiencies:** The project, as mentioned, was cluttered with useless code that was not necessary and simply took longer than necessary to run the code.

All of these issues compounded to make Conical Slicing not as big as it should have been. I have spent several months learning and fixing the problems to present Conical Slicing in the light it deserves.

# 4. Process

Conical Slicing is a solution to the problem of overhangs. Instead of building it up layer by layer, it builds it up shell by shell. These shells end up just being angled versions of the aforementioned layers. The process involved three main steps.

The first step is called transformation. First, we put a STL file, which we just established as a formation of triangles, into a Python program. The first thing it does is refine the STL file. This entails taking the midpoints of all the lines of each triangle and making them new points; then, you connect each new point to each other. This results in 4 triangles instead of 1, this is repeated as many times as inputted and is important for the next step. Then, each point on each triangle is transformed or, in other words, moved upward proportional to its distance to the origin (Figure 4). The math used to find the exact amount is explored further in the math section. This is where we can see how important it is that we complete the refinement step so that the transformed model will have a clean cone shape and complete the process successfully.

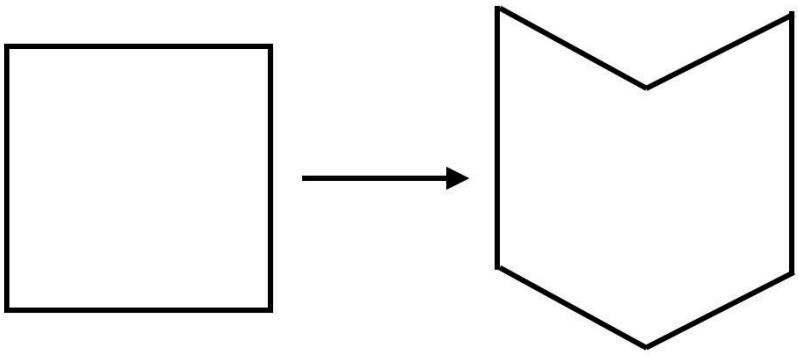

Figure 4
A drawing of the
transformation process

The second step is to simply put the transformed STL file into a slicer. This is a very important part as this is what turns the STL into G-code. It may seem confusing why you would use the one thing you are trying to improve upon, but this is simply unavoidable. Modern 3D printing slicers are so advanced and good at their job that it is simply easier just to transform a model and later back-transform it rather than make a whole new slicing application with way more problems and advanced math. It is also imperative to change some of the settings that I talk about later, so the next step can happen smoothly.

The third and final step is to back-transform the G-code file. This entails sending the G-code file back into a Python program which goes through each of the tens if not hundreds of thousands of lines and moves the z for every point downward based on its distance from the origin in the opposite way as the transformation step (Figure 5). But first, because of the way that the slicers work, we have to move every point from the center of the build plate to be centred around the origin, usually just half the length and width of the build plate. It is important that you move it the correct distance, so some trial and error is necessary to dial it in. One thing to know is that if you have it wrong, you can tell as the back-transformation will be skewed. Then once the model is back-transformed, we move it back to the center in the same way as before.

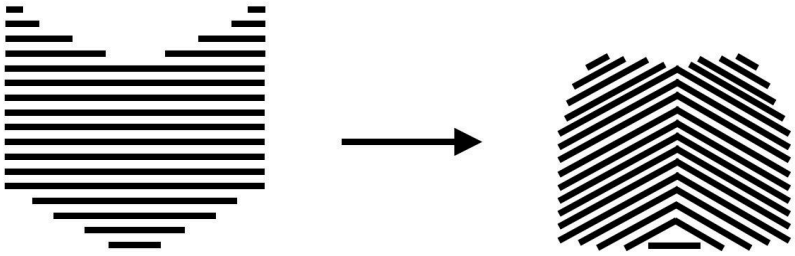

Figure 5
A drawing of the
back-transformation process

# 5. Math

As I stated in the process section for Conical Slicing we have to transform or move the points proportional to its distance to the center. To do this, it is easiest to start by making this 3D problem into two dimensions. To do that, we have to find the distance to the center, the formula for that is:

$$a = \sqrt{x^2 + y^2}$$

In this equation, a is short for adjacent; this is important later. Also x and y are the x and y coordinates of the points.
After we find the distance to the origin, we can imagine a triangle with the origin as one point, the point we want to transform as another, and where those two meet as a third. Then we can think of this as a trigonometric question and use the formula

$tan(\theta) = \frac{o}{a}$ {The usual state of the equation}
$tan(\theta) * a = o$ {The state necessary for my purposes}

In the proper formula shown above, we can see that if we have the angle of the triangle or the cone, and the distance we just found, we find the amount we need to move the point. The one thing to remember is to turn the cone angle for degrees into radians for the formula.

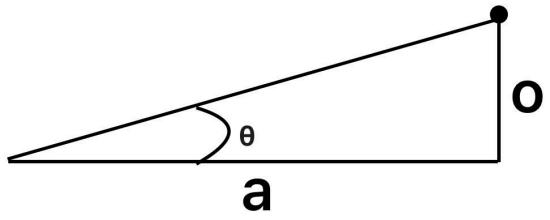


Figure 6
A visual representation of the math, where the black point is the point we have.

# 6. Final Product

Throughout this report, I have shared lots about the concept of conical slicing, but now I get to share what I have actually done.

Firstly, I would like to summarize what my project can do. It can successfully carry out both inward and outward cones for Conical Slicing (Figure ?). For context, the difference between inward and outward-facing overhangs is whether they face toward or away from the centre. It can also do a more advanced technique of stacking G-code to have both inward and outward angles in one part, although that is not the focus of this project. My project also has made Conical Slicing work with the two most popular 3D printing slicers, Cura and Prusa slicer. This is especially important, as almost all slicers are based on one of these two, so in effect, I brought Conical Slicing to the mass market.

Secondly, I have made the code significantly more efficient. In fact, I have improved the transformation process by 30% and decreased the time taken for back-transformation by a whopping 20 times.

Thirdly, I have decreased the hassle of keeping track of so many documents and made it so a person would only need to have one file and to interact through the command prompt. This significantly reduces the struggle of the files I had to work with originally. Along with this, I created a setup and usage guide. The code and these are both available on my GitHub at https://github.com/Ezra-Huska/conical-slicing.

Fourthly, I have made an updated list of crucial settings to change in the slicer:

- Where you have your starter G-code, remove the primer line, and also put the line G90 in to make sure it uses absolute coordinates
- Set the bottom to 0 layers
- If you are using a Prusa slicer, pick an infill with curved lines, not straight. If you are using Cura, you have more leniency.
- Change your travel settings to ramping lift and the cone angle you want to slice at in degrees.
- Slow down the printing speed, I have done 20mm/s for infill and 15mm/s for walls and top.

All of these are very important in their own way to make sure it works properly and does not wear down your nozzle.

# 7. Conclusion

As I have mentioned in my introduction, 3D printing has been a large upgrade over its predecessor, injection moulding. Through this project, I have explored a mathematical solution to the problem of overhangs, one of the current challenges with 3D printing methods.

The first of these improvements came in the form of making Conical Slicing work with the two most popular 3D printing slicers and by extension, over one million people worldwide. The second of these was to significantly improve the speed of the programs needed to compute it, in some cases by over 2000%. The third was to simply make it easier to use by making it better documented and simpler to use for beginners.

# 8. Applications

The applications extend far further than the hobby 3D printing market. It can extend to the greater engineering context, product development, and even manufacturing.

The effects on the greater engineering context are significant.. Conical Slicing is a massive improvement over support material. It reduces the time required to remove supports and increases the capabilities of 3D printers. It allows for the printing of parts thought to be impossible based on their geometry and opens the doors to new innovations.

The effects on the greater engineering context are significant as mentioned before, Conical Slicing is a massive improvement over support material. This is especially crucial in this field as iIt reduces the time required to remove supports and increases the capabilities of 3D printers. It allows for the printing of parts thought to be impossible based on their geometry, and that opens the doors to new innovations.

Conical Slicing opens the door to faster and better product development and manufacturing. It gives us the ability to print more complex geometries,  making it a more useful tool in the realm of product development. Conical Slicing is the start of a revolution.  Using more 3D printing in the development process brings hundreds of new, more advanced products and solutions to problems. This will also have a massive effect on the manufacturing industry. In the past, 3D printing has been inconsistent, wasting plastic and prone to failure. Too often, these problems occur because the 3D printer is unable to print overhangs. This project solves that problem and will open up the mass production of more complex parts.

# 9. Further Development

Conical Slicing, in its current state, is not a perfect all-encompassing solution to the problem of overhangs. As seen before, you can have either inward or outward-facing cones, and even sometimes on the same part. But what you can't have is both cone types at the same time or at the same layer. This means that some parts are still unprintable, and some parts are worse off than before, but this is not the way it has to be. There are many ways to move forward with the development of this technology.

First, this project is not simply an end but a beginning of the development of a universal slicer. This idea of a universal slicer is a program that you put in a 3D model, and somehow, through a mixture of angled slicing, and G-code bending, the part has no overhangs that it can't print. I believe this dream will very soon become a reality.

The second way to continue the development of this technology is to make it more accessible. Currently, to do conical slicing, you have to run at least one Python file, which means you must have basic programming knowledge and have Python installed on your computer. This could be significantly improved into just a toggle in the slicer you already use. It is important for slicer developers to hear about the technology and know how to implement it properly.

Thirdly, in order to take advantage of Conical Slicing, 3D printer manufacturers need to design better fan mounts that allow for greater clearance around the nozzle and, therefore, higher cone angles, leading to an even better ability to print more overhangs. This will lead to more people taking advantage of Conical Slicing and lead to other developments from the community.

# 10. Acknowledgments

Firstly, I would like to express my sincere gratitude to all of those who have helped me through helping solve the problems in my code. This includes my parents, my friends and my computer science teacher Mr. Marshal. I would also like to thank my mom for helping me prepare my backboard and my dad for proofreading my information. Finally, I would like to thank all the people who have kept me motivated and encouraged throughout this whole project.

# 11. References

Hermann, S. (n.d.). *Conical Slicing: A different angle of 3D printing*. CNC Kitchen. https://www.cnckitchen.com/blog/conical-slicing-a-different-angle-of-3d-printing

Hermann, S. (2022, November 11). *How To Use Conical Slicing*. CNC Kitchen. https://www.cnckitchen.com/blog/guide-how-to-use-conical-slicing

Wüthrich, M., Gubser, M., Elspass, W. J., & Jaeger, C. R. (2021). A Novel Slicing Strategy to Print Overhangs without Support Material. *MDPI*, *11*(18), 8760–8760. https://doi.org/10.3390/app11188760

Google. (2024). Gemini (2.0 Flash) [Large language model]. https://gemini.google.com

Chapman, A. (2022, July 29). *The complete history of 3D printing*. UltiMaker. https://ultimaker.com/learn/the-complete-history-of-3d-printing/

Joshi, S. (2025). *75+ 3D Printing Statistics and Trends to Follow in 2025*. G2.com. https://learn.g2.com/3d-printing-statistics/

# 12. Appendix

# Setup and Usage Guide

This is a guide meant for people who have little to no experience in coding or 3D printing.

# Set up

1. Download python from:
2. Go to the GitHub at https://github.com/Ezra-Huska/conical-slicing, and copy the code the from the Main.py into a file of that name on your computer
3. Go to your command prompt and type in python -m pip install STL, and then let it finish downloading.

Simple as that it is set up.

# Usage

## 1. Transform

1. Open up your command prompt and follow this formula, python3 “path/file_name.py” and click enter.
2. Now type tf for transformation, the first step in the Conical Slicing process.
3. Now look at the 3d model you want to transform, and see if the overhangs are inward, or outward, and type that in the next spot.

4. Now decide how many refinements you want, usually 2 or 3 is good, but the more you do, the longer it will take.
5. Next, input the file path to the STL you want to transform and put it there.
6. Input the location of the folder of where you want the transformed file to be stored.
7. Now wait for it to be done.

## 2. Slice

1. Now put the transformed model into your slicer of cho, and slicer the slice model. I recommend changing these settings first:
    - Where you have your starter G-code, remove the primer line, and also put the line G90 in to make sure it uses absolute coordinates
    - Set the bottom to 0 layers
    - If you are using Prusa slicer, pick an infill with curved lines, not straight. If you are using Cura, you have more leniency.
    - Change your travel settings to ramping lift and the cone angle you want to slice at in degrees.
    - Slow down the printing speed, I have done 20mm/s for infill and 15mm/s for walls and top.

## 3. Back-Transform

1. Next, open up your command prompt again and type what you did at the start of the transformation step.
2. Now type bt for back-transformation.
3. Then type the same cone type you used for the transformation process.
4. Also, say the same cone angle.
5. Next, input the file path and the file of the G-code file you want to back-transform, and put it there.
6. Input the location of the folder where you want the back-transformed file to be stored.
7. After that say half the size of your printing area of the printer. First, the x, then next, the y. If you used cura for the slicing, try removing 5 from each of the x and the y.
8. Then just wait for it to be done.

I recommend you try to dial these settings in before printing for you first time, and if it doesn’t look right, it probably isn't right. Also, if you are more advanced, instead of saying cone type, you can write default. I have hand coded in my personal settings, and you can change them and save time later. Also, if you want and are more advanced, you can use two separate files for transformation and back-transformation.