

Extracting Slicer Parameters from STL file in 3D Printing

Sonali Patil¹, Yogesh Deshpande², Dattatraya Parle³

Submitted: 10/12/2023 Revised: 21/01/2024 Accepted: 31/01/2024

Abstract: These 3D printing revolutionizes manufacturing by requiring accurate control and optimization of print parameters. Slicer software simplifies 3D modelling for printers by breaking down models into layer-wise instructions, calculates toolpath, generates support structures, aids in infill density, pattern control, and sets print settings. It includes a 3D preview and uses G-code, a 3D printing language, for printer setup. Extracting slicer parameters from G-code is crucial for quality control, documentation, optimization, troubleshooting, and educational purposes. The feature allows users to review settings and parameters during the slicing process, improving quality control, facilitating troubleshooting, identifying improvement areas, reducing print times, and enhancing material efficiency. Analysing slicer parameters in G-code can offer valuable insights into the printing process, enabling fine-tuning of print settings for enhanced quality and efficiency. This research paper reviews the challenges and techniques of extracting slicer parameters like G-code Parsing Algorithms, Regular Expression Matching, Metadata Extraction and Machine Learning Approaches. A novel method is discussed to extract the features by parsing the sliced STL file. The dataset generated can be further used to find Layer Thickness, Layer Height Distribution, Surface Quality, Interlayer Adhesion, Slice Alignment, Defect Detection and Geometric Analysis.

Keywords: G-code, STL, Slicer Parameters, 3D Printing, Parameter Extraction, G-code Parsing, Slicer Software, Geometrical Feature Extraction, Slicing Algorithms, Parameter Optimization

1. Introduction

3D printing is a process that creates three-dimensional objects by material layer on layers. 3D printing has emerged as a transformative technology with applications spanning various industries, including manufacturing, healthcare, and design [1]. The process involves the creation of three-dimensional objects by depositing successive layers of material based on a digital model [2]. Slicer software plays a critical role in the 3D printing workflow, converting 3D models into machine-readable G-code instructions that guide the printer's movements and extrusion [3].

Achieving optimal print quality and efficiency in 3D printing requires precise control of slicer parameters. These parameters, including layer height, infill density, print speed, and material settings, constitute pivotal factors that profoundly influence the final output in additive manufacturing [4], [5], [6]. The layer height determines the resolution of the printed object, affecting its surface finish and overall accuracy [7], [8]. Infill density directly correlates with the internal structure and strength of the printed item [9], [10]. Print speed affects the rate at which material is deposited, influencing not only production time but also heat distribution and layer adhesion [11],[12]. Material settings encompass variables such as temperature and flow rate, significantly affecting the material's behavior during the printing process and, consequently, the mechanical

properties of the final product [13],[14]. These parameters collectively play a crucial role in defining the quality, structural integrity, and functional characteristics of 3D-printed objects.

However, accessing and understanding the slicer parameters embedded within the G-code can be challenging, as the G-code files are often complex and lack human-readable annotations. Extracting and analyzing these parameters can provide valuable insights into the printing process, enable parameter optimization, and facilitate printer calibration [15].

1.1. Research Objectives:

The primary objectives of this research paper are as follows:

- To investigate the techniques for extracting slicer parameters from G-code in 3D printing.
- To explore the importance and impact of slicer parameters on print quality, print speed, material usage, and support structures.
- To analyze the challenges associated with extracting slicer parameters, such as complex G-code structures, parameter variations across slicers, limited documentation, and scalability
- To compare and evaluate different techniques for slicer parameter extraction, including G-code parsing algorithms, regular expression matching, metadata extraction, and machine learning approaches.

^{1,2}Department of Computer Engineering, Vishwakarma University, Pune, India;

³Nuclear Advanced Manufacturing Research Centre, Sheffield, United Kingdom

e) To demonstrate the practical applicability of slicer parameter extraction through sliced STL (Standard Tessellation Language) file using python.

f) To identify future research directions for automating parameter extraction, exploring advanced machine learning techniques, and integrating parameter extraction within slicer software.

By addressing these research objectives, this paper aims to contribute to the advancement of 3D printing technology by enabling a better understanding and utilization of slicer parameters for improved print outcomes and enhanced efficiency.

2. Slicer Software and G-code

Slicer Software Overview:

[1], [16], [17] highlight the critical component of slicer software in 3D printing workflows. Ultimaker Cura, PrusaSlicer, Simplify3D, and Slic3r are popular slicer software options, each with unique features and algorithms. These software options enable users to customize the printing process, generate toolpaths, and add support structures.

G-code Structure and Representation:

Fig.1 shows the G-Code structure. G-code is a language used to control computer numerical control (CNC) machines, including 3D printers. It consists of a series of commands that instruct the printer on how to move, extrude material, and perform various other actions during the printing process. G-code files are plain text files that can be opened and edited using any text editor [3]. The structure of G-code is based on a series of commands, where each command begins with a specific letter, known as the command code, followed by parameters or values associated with that command. For example, the command "G1 X100 Y50 Z10 F300" instructs the printer to move the print head to the position X=100, Y=50, and Z=10 at a feed rate of 300 mm/min [3].



Fig 1. G-Code structure

Slicer Parameters in G-code:

Slicer parameters are essential settings that determine various aspects of the printing process and are embedded within the G-code. These parameters are derived from the user-defined settings in the slicer software and are translated into corresponding G-code commands.

Common slicer parameters found in G-code include:

- a) Layer Height: Specifies the vertical thickness of each printed layer.
- b) Print Speed: Determines the speed at which the print head moves during printing.
- c) Infill Density: Controls the amount of infill material within the printed object.
- d) Extrusion Temperature: Sets the temperature of the extruder nozzle for melting the filament.
- e) Retraction Settings: Manages the retraction and unretraction of filament to prevent stringing and oozing.
- f) Support Structures: Specifies the generation of additional structures to provide support during printing.
- g) Cooling Settings: Regulates the cooling fan speed to improve print quality and reduce warping [3], [15].

These slicer parameters influence crucial aspects of the printing process, such as print quality, printing time, material consumption, and structural integrity. Extracting slicer parameters from G-code involves parsing the G-code file, identifying the relevant commands, and extracting the associated parameter values.

The extracted parameters can then be analyzed and utilized for various purposes, including optimization, quality assessment, and process control in 3D printing. Table 1 shows the effect of slicer parameters on printed part. In the following sections, we will explore the techniques and challenges involved in the extraction of slicer parameters from G-code in 3D printing.

3. Importance of Slicer Parameters

3.1 Impact of slicer parameters on printed part

The quality and efficiency of 3D printed objects heavily rely on the selection and optimization of slicer parameters. Slicer parameters, such as print quality, print speed, material settings, support structures, layer heights, and infill density, significantly affect the final output. This research article examines the importance of slicer parameters in 3D printing and investigates their influence on various aspects of the printing process. Table 1 shows Effect of slicer parameters on printed part. With the increasing popularity and adoption of 3D printing technology, it becomes crucial to understand the role of slicer parameters in achieving high-quality prints. Slicer parameters directly affect the printing process and can greatly affect print quality, speed, material usage, and structural integrity. Optimizing slicer parameters is essential for obtaining desired print outcomes, minimizing defects, reducing printing time, and maximizing material efficiency. By exploring the importance of slicer parameters, this research aims to provide guidance to 3D printing professionals.

3.2 Impact of slicer parameters in printing scenarios

The table 2 provides a concise overview of how slicer parameters affects the 3D printing process and highlights their applications across different industries and printing scenarios.

3.3 Relationship between slicer parameters and printing aspects

Table 3 provides a detailed analysis of how slicer parameters are interconnected with different aspects of the 3D printing process, highlighting their relationships and impacts on the final print. Concisely below are the insights of impacts of slicer parameters in printing scenarios.

Table 1: Effect of Slicer Parameters on Printed Part

Slicer Parameter	Effect on Printed Part	References
Layer Height	Influences surface finish and detail resolution.	[31], [32], [33]
Print Speed	Affects print time and can affect print quality.	[32], [33],[34]
Infill Density	Influences part strength and material usage.	[10],[31], [32]
Support Structures	Necessary for overhangs, affects surface quality	[31], [32], [34]
Brim/Raft	Enhances adhesion to the build plate.	[31], [33], [34]
Print Temperature	Affects material flow and layer adhesion.	[33], [34]
Bed Temperature	Influences adhesion and material properties.	[34], [35]
Retraction Settings	Helps reduce stringing between moves.	[33], [34]

Table 2: Impact of Slicer Parameters in Printing Scenarios

Slicer Parameters	Importance	Application in Industries/Scenarios
Layer Height	Determines vertical resolution; affects print time.	Aerospace (precision components), Medical (surgical models), Prototyping
Print Speed	Affects print time and quality.	Rapid prototyping, High-precision components
Infill Density	Influences print strength and time.	Manufacturing (strong components), Decorative items
Support Structures	Essential for overhangs and complex geometries.	Architecture (intricate designs), Art (complex sculptures)
Temperature Settings	Affects layer adhesion and print quality.	Medical (biocompatible materials), Electronics (specific profiles)
Bed Adhesion	Crucial for preventing warping and ensuring adhesion.	Electronics manufacturing, Consumer goods production
Cooling Settings	Prevents overheating, improves print quality.	Electronics (precise cooling for components)
Bridging Settings	Influences the printer's ability to span gaps.	Architectural models, Engineering prototypes

Table 3: Relationship Between Slicer Parameters and Printing Aspects


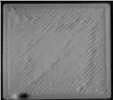




Aspects of 3D Printing	Slicer Parameters	Relationship and Impact
Print Quality	- Layer height	Smaller layer heights improve detail but increase print time.
	- Temperature settings	Proper temperature enhances layer adhesion and overall print quality.
Print Speed and Efficiency	- Print speed	Faster speed reduces print time but may sacrifice quality.
	- Infill density	Balancing density and speed is crucial for efficiency.
Structural Integrity	- Infill density	Higher infill provides internal support and enhances structural integrity.
	- Support structures	Proper supports ensure the integrity of complex geometries.
Adhesion and Warping	- Bed adhesion	Proper adhesion settings prevent warping and ensure successful prints.
	- Cooling settings	Controlled cooling prevents overheating and warping.
Complex Geometries and Overhangs	- Support structures	Customizable supports enable successful printing of intricate designs.
	- Bridging settings	Parameters influence the printer's ability to bridge gaps without supports.
Precision and Detail	- Layer height	Smaller layer heights enhance precision and detail but extend print time.
	- Print Speed	Slower speeds are chosen for high precision and fine details.

The cited literature emphasizes the pivotal role of slicer parameters in 3D printing, with a focus on layer height, infill

density, support structures, print speed, extrusion temperature, retraction settings, material flow rate, and their intricate interplay. Smaller layer heights contribute to finer details and smoother surfaces, affecting print resolution and overall quality [18]. Infill density influences strength, weight, and structural integrity, with higher densities enhancing robustness but potentially extending printing time [19]. Support structures, crucial for overhangs and complex geometries, benefit from optimization in density and pattern to improve print quality [20]. Print speed affects accuracy, surface quality, and dimensional precision, with higher speeds potentially compromising quality [21]. Extrusion temperature directly influences filament flow and layer adhesion, and its optimization improves print quality [20]. Retraction settings, material flow rate, and layer heights significantly affect print quality, mechanical strength, and printing time. The research article aims to delve into these parameters, providing detailed analysis, experimental results, and practical recommendations for their optimization in 3D printing

Table 4 shows the errors occurred when incorrect slicer parameters are considered.

Table 4: Issues during printing and solution

Study	Part 1	Part 2	Part 3
Issue	 Stringing And Oozing	 First Layer Issues	 Layer shifting
Solutions	 <ul style="list-style-type: none"> Adjusted retraction distance and speed Optimized print temperature 	 <ul style="list-style-type: none"> Used appropriate bed adhesion Adjusted the nozzle height during bed leveling Set the nozzle & bed temperature 	 <ul style="list-style-type: none"> Ensure Proper Cooling Printed at Slower Speeds

Future research in the importance of slicer parameters: The importance of slicer parameters in 3D printing continues to be a subject of active research as the technology evolves. Future research in this field may explore various aspects to enhance the capabilities, efficiency, and versatility of 3D printing. Some potential areas of future research in the importance of slicer parameters include Machine Learning and Adaptive Slicing, Automated Optimization Tools, Process Optimization for Industrial Applications, Real-Time Monitoring, Control, and Standardization of Slicer Parameters.

4. Techniques for Slicer Parameter Extraction

4.1 G-code Parsing Algorithms:

G-code, being the language that controls 3D printers, is inherently text-based and requires parsing to extract meaningful information. G-code parsing algorithms involve breaking down the G-code instructions into structured data, enabling the extraction of specific parameters. Techniques such as lexical analysis and syntax parsing are commonly

employed [6], [22]. These algorithms are foundational for understanding the toolpath, printing speeds, and other critical settings embedded in the G-code. G-code parsing algorithms are an integral part of the process for extracting slicer parameters from G-code files in 3D printing. These algorithms are designed to analyze the structure and syntax of the G-code and extract relevant information related to slicer settings and parameters.

The complexity of G-code files makes it challenging to identify and extract specific slicer parameters. G-code commands can include movement instructions, extrusion commands, temperature settings, and other control codes. Furthermore, G-code commands can be nested, contain conditional statements, or employ loops, making it necessary to develop robust algorithms to accurately traverse and interpret the code structure. Several parsing algorithms have been proposed in the literature to address these challenges. These algorithms utilize various techniques to extract slicer parameters, including tokenization, pattern matching, command sequence analysis, and syntax tree construction. Tokenization involves breaking down the G-code file into individual tokens or meaningful units, such as command codes, parameters, or comments. Tokens are then analyzed to identify relevant slicer parameters. For example, a token may represent a movement command (e.g., G0 or G1) or an extrusion command (e.g., M101 or M103), and the associated parameter values can be extracted from these tokens.

Pattern matching techniques utilize regular expressions or predefined patterns to identify specific codes or patterns within the G-code file. These patterns are designed to capture slicer-related commands or parameters, allowing for their extraction. Regular expression libraries, such as Python's re module, are commonly used for this purpose. Command sequence analysis involves examining the sequence of G-code commands to identify specific patterns or combinations that represent slicer parameters. For instance, consecutive commands related to extrusion or temperature control can indicate relevant slicer settings.

Syntax tree construction involves building a hierarchical representation of the G-code commands and their relationships. By constructing a syntax tree, it becomes easier to navigate through the G-code structure and extract the desired slicer parameters. G-code parsing algorithms involve analyzing the structure and syntax of the G-code file to extract relevant slicer parameters. These algorithms typically involve traversing the G-code commands, identifying specific codes associated with slicer parameters, and extracting the corresponding parameter values. Different parsing techniques, such as tokenization, pattern matching, and command sequence analysis, can be employed for effective parameter extraction [23]. By

employing these parsing algorithms, researchers and practitioners can effectively extract slicer parameters from G-code files, enabling further analysis, optimization, and control of the 3D printing process.

4.2 Regular Expression Matching:

Regular expression matching is a powerful technique for extracting structured information from text. In the context of slicer parameter extraction, regular expressions can be tailored to recognize specific patterns indicative of parameters within G-code or STL files. This method allows for a more flexible and customizable approach to parameter extraction, accommodating variations in G-code formats [24], [10]. Regular expression matching is a powerful technique used in the extraction of slicer parameters from G-code files in 3D printing. It involves defining patterns using regular expressions to identify specific command codes or parameter values within the G-code. By matching these patterns, relevant parameter values can be extracted efficiently. Regular expressions are a sequence of characters that define a search pattern. They provide a concise and flexible way to match and extract specific text patterns from a larger body of text, such as a G-code file. Regular expressions can be used to identify command codes (e.g., G0, G1) or parameter values (e.g., federate, extrusion rate) associated with slicer settings.

Regular expression patterns combine literal characters and metacharacters for matching rules. Metacharacters like dot, asterisk, and square brackets allow for patterns, repetition, or ranges. Libraries like Python's re module offer functions for pattern matching, substitution, and group extraction. Regular expressions are useful in slicer parameter extraction, as they can handle syntax variations and structure differences in G-code files. Regular expression matching is a powerful technique for extracting slicer parameters from G-code. It involves defining patterns using regular expressions to identify specific command codes or parameter values within the G-code file. By matching these patterns, relevant parameter values can be extracted efficiently. Regular expression libraries and tools, such as Python's re module, are commonly used for this purpose [25]. By utilizing regular expression matching techniques, researchers and practitioners can effectively identify and extract relevant slicer parameters from G-code files, enabling further analysis, optimization, and control of the 3D printing process.

4.3 Metadata Extraction:

Metadata extraction involves retrieving information embedded within the file headers or other designated sections. In G-code and STL files, metadata can contain valuable details about the print job, including layer heights, temperatures, and print speeds. Extraction methods often

include algorithms to locate and interpret metadata tags [9], [26]. This technique is particularly useful for obtaining global settings applied throughout the print. Metadata extraction in slicer parameter analysis involves accessing embedded information, often comments or annotations, within G-code files. These annotations, commonly added by users in slicer software, offer insights into specific parameters like layer heights or print speeds. By extracting and interpreting this metadata, researchers gain a deeper understanding of slicer settings beyond what's explicitly in G-code commands, complementing traditional extraction methods. Metadata tags in G-code, such as "; Layer Height:", serve as guides for extraction, enhancing context and providing additional details for a comprehensive analysis of slicer parameters.

The work by [25] discusses the application of metadata extraction for G-code analysis in 3D printers. The authors propose a method that includes the extraction of slicer parameters from metadata tags embedded in the G-code. By parsing the comments and annotations, the authors extract additional information related to print speed, extrusion rates, and other slicing settings.

4.4 Machine Learning Approaches:

Machine learning approaches are gaining prominence in slicer parameter extraction due to their adaptability and ability to discern complex patterns. Supervised learning models, such as regression or classification algorithms, can be trained on labeled datasets to predict slicer parameters directly from G-code or STL features. Convolutional Neural Networks (CNNs) have shown efficacy in image-based parameter extraction, while recurrent neural networks (RNNs) are adept at sequential data analysis, making them suitable for G-code parsing [11],[27]. Machine learning offers a promising approach for extracting slicer parameters from 3D printing G-code files. Supervised learning algorithms like support vector machines, random forests, and deep learning architectures, such as convolutional and recurrent neural networks, demonstrate effectiveness in recognizing patterns and dependencies between G-code commands and slicer parameters. These methods adapt to diverse G-code structures and variations in slicer software outputs, automatically learning relationships without predefined rules. Deep learning models like CNNs capture spatial patterns, while RNNs handle temporal dependencies, achieving high accuracy through labeled dataset training. Additionally, reinforcement learning techniques can optimize parameter extraction by iteratively adjusting strategies based on feedback and rewards, showcasing the versatility of machine learning in enhancing the efficiency and adaptability of slicer parameter extraction

One example of the application of machine learning for G-code analysis and slicer parameter extraction is presented in

the work by [28]. The authors propose a deep learning-based approach that utilizes CNNs for G-code analysis and extraction of slicing parameters for additive manufacturing. Their model effectively captures the relationships between G-code commands and slicer parameters, achieving accurate parameter extraction.

By leveraging machine learning approaches, researchers can enhance the accuracy and efficiency of slicer parameter extraction from G-code files. These approaches provide a data-driven and adaptive approach to parameter extraction, enabling more advanced analysis, optimization, and control of the 3D printing process. Machine learning approaches offer promising avenues for slicer parameter extraction from G-code. These techniques involve training models using labeled datasets of G-code files and associated parameter values. The models can learn to recognize patterns, correlations, and dependencies between G-code commands and the slicer parameters they represent. Supervised learning algorithms, such as support vector machines (SVM), random forests, or deep learning architectures, can be applied for parameter extraction [28]. The choice of machine learning approach depends on the specific problem and available data. Convolutional neural networks (CNNs) have shown promising results in analyzing the structure and content of G-code files, enabling the extraction of slicer parameters [28]. Reinforcement learning techniques can also be explored to optimize the extraction process based on feedback and rewards [29].

The issue of inadequate labeled data and complex parameter relationships hinders optimal parameter extraction performance. Continuous advancements in machine learning focus on data accessibility, model interpretability, generalization, and domain-specific knowledge integration. The integration of extraction techniques into 3D printing workflows involves machine learning modules, regular expression matching, and G-code parsing algorithms. These techniques optimize slicer parameters based on historical

print data, enabling real-time adjustments to suggested parameters, enhancing efficiency and user-friendliness in the slicing process

5. Challenges in Slicer Parameter Extraction

5.1 Complex G-code Structures:

The paper [23] highlight the complex structure of G-code commands, which can be nested, conditional statements, or loops, making it challenging to extract slicer parameters. This complexity impacts 3D printing technologies and materials, necessitating robust parsing algorithms and machine learning for G-code optimization to accurately navigate and interpret the structure.

5.2 Parameter Variations across Slicers:

The work [23] highlight the need for parameter mappings to ensure consistency across different slicer software and G-code files, addressing the challenge of extracting parameters due to the diversity of slicer software outputs.

These examples from table 6 highlight how parameter variations across slicers can manifest in critical aspects of the printing process. Strategies to address these inconsistencies include standardization initiatives, profile sharing platforms, translation tools, user education, and harmonization tools, collectively aiming to enhance consistency and user experience in 3D printing.

5.3 Limited Documentation:

The research [25] highlights the limited documentation for slicer parameters in G-code files, making it difficult to accurately extract them. This lack of comprehensive information can lead to manual exploration and trial and error. Addressing this requires a multifaceted approach involving community and stakeholders for a robust knowledge-sharing ecosystem.

Table 5.: Machine learning in 3D printing

Criteria	SVM	CNN	RNN
Nature of Technique	Linear classifier suitable for simpler tasks.	Specialized for image data, capturing spatial hierarchies.	Sequential data processing, suitable for time-series or sequential data.
Application in 3D Printing	Quality control, fault detection.	Image-based tasks (e.g., object recognition, segmentation).	Sequential data tasks (e.g., predicting print time).
Computational Efficiency	Efficient for smaller datasets and linear tasks.	Computationally intensive, especially for large datasets.	Depends on the complexity of the model; can be resource-intensive.
Robustness	Robust for well-defined tasks with clear boundaries.	Robust for image-related tasks with abundant data.	Sensitive to sequence length and may struggle with long-range dependencies.
Data Requirements	Less data-hungry for linear tasks.	Requires substantial labeled image data for training.	Requires labeled sequential data, might need substantial training data.

Table 6: Parameter Variations across Slicers

Parameter	Manifestation of Variations	Strategies to Address
Layer Height	Inconsistent layer thickness, affecting print resolution and surface finish.	Standardization initiatives for common layer height values.
Infill Density	Variations in infill patterns and density impact print strength & weight.	Profile sharing platforms for standardized infill settings.
Print Speed	Differences in print speed affect printing time and structural integrity.	Translation tools converting speed values between slicers.
Temperature Settings	Variances in temperature can lead to material adhesion and print quality issues.	User education on manual adjustment and optimization for different slicers.
Support Structures	Inconsistencies in support generation impact ease of removal and print stability.	Harmonization tools analyzing and aligning support structures.

5.4 Scalability:

The work [28] highlights the importance of scalability in extracting G-code structures from complex data. As 3D printing technologies advance, the complexity of G-code structures increases, necessitating efficient algorithms and optimization strategies to handle large-scale extraction tasks. These techniques must account for parameter variations across slicers, overcome limited documentation, and ensure practical applicability in real-world scenarios, ensuring the practical applicability of slicer parameter extraction methods. Research is ongoing to develop algorithms and architectures to efficiently handle large datasets in the 3D printing community. Techniques such as parallel processing and distributed computing are being explored to improve the speed and efficiency of parameter extraction algorithms [22], [30]. This is crucial for real-time feedback and adaptation in automated manufacturing environments. The research aims to create solutions that can adapt to the increasing complexity and diversity in 3D printing applications.

5.5 Interdependencies of Parameters:

Extraction algorithms face challenges due to interdependencies in slicer parameters, impacting the accuracy of the printed object [17],[19]. These issues have significant implications for the industry, including scalability, documentation, and complex G-code structures. Addressing these issues enhances operational efficiency, reduces costs, and promotes sustainability. Future research in 3D printing aims to advance machine learning algorithms for dynamic adaptability and predictive modeling. Parameter extraction may involve automated inference and material science principles. Challenges include parsing G-code, regular expression matching, metadata extraction, and standardization. Machine learning faces limitations in

labeled datasets, leading to data extraction through STL files.

6. Experimental Methodology: Generating a Reference Dataset using STL File

Application of different slicer parameter techniques like G-code Parsing Algorithms, Regular Expression Matching, Metadata Extraction, Machine Learning Approaches and also addressing technical challenges in additive manufacturing, including scalability, limited documentation, parameter variations, interdependencies of parameters, and complex G-code structure, a novel approach is proposed to extract the printing coordinates of each layer from STL file. Mass properties can be extracted from STL file using python.

6.1 To visualize an STL file using Python

This section shows how to visualize STL (STereoLithography) files using Python. STL files are widely used in computer-aided design (CAD) and 3D printing as a common format for representing 3D models. Python provides several libraries that facilitate loading, processing, and visualizing STL files.

Importing Libraries:

> The first step is to import the necessary libraries. In this case, matplotlib.pyplot used for creating the visualization and mpl_toolkits.mplot3d for enabling 3D plotting.

Loading the STL File:

> The code loads the STL file using the appropriate library or method. This could be accomplished using different libraries such as numpy-stl, meshio, or specialized libraries like trimesh. The specific library and method used may vary depending on the requirements and preferences of the user.

Extracting Vertices:

- > Once the STL file is loaded, the next step involves extracting the vertices of the 3D model. These vertices define the shape and structure of the model.
- > The code accesses the vertices from the loaded STL file and stores them in separate arrays for x, y, and z coordinates.

Creating a 3D Plot:

- > After extracting the vertices, the code creates a 3D plot using `matplotlib.pyplot`.
- > The `fig = plt.figure()` statement initializes a new figure to which the plot will be added.
- > The `ax = fig.add_subplot(111, projection='3d')` line creates an `Axes3D` object, enabling 3D plotting on the figure.
- > The `ax.scatter()` function is then used to plot the extracted vertices as points in the 3D space.

Customizing the Plot:

- > The code allows for further customization of the plot. This includes adding labels and a title to the axes, changing the color and marker style of the plotted points, adjusting the aspect ratio, and setting limits for the axes.

Displaying the Plot:

Finally, the `plt.show()` function is used to display the 3D plot. Conclusion: Visualizing STL files in Python can be achieved using the `matplotlib` library in combination with other specialized libraries for loading and processing STL files. The code provides a basic framework for loading an STL file, extracting the vertices, creating a 3D plot, and customizing the visualization.

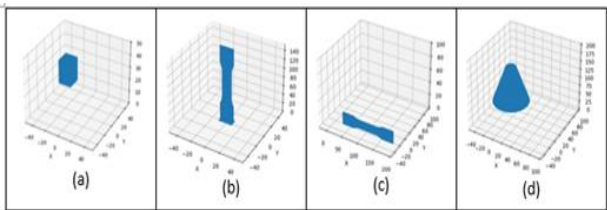


Fig 2 STL file visualization of a) Cube b) Upright Model c) Model at Edge d) Frustum

6.2 Slicing and Visualizing STL Files in Python

This section discusses the code and steps involved in slicing and visualizing STL files using Python. STL (Standard Tessellation Language) is a file format commonly used for 3D printing and computer-aided design. Slicing an STL file involves extracting specific layers or sections from the 3D model and visualizing them to gain insights or prepare them for further analysis.

The code for slicing and visualizing STL files can be divided into several steps:

1. Importing Required Libraries: As `numpy`, `matplotlib`, `stl`, `mpl_toolkits.mplot3d`.
2. Loading the STL File: Use the `stl.mesh.Mesh.from_file()` function to load the STL file.
3. Visualization: Create a 3D plot using `matplotlib.pyplot.figure()` and `mpl_toolkits.mplot3d.Axes3D`.

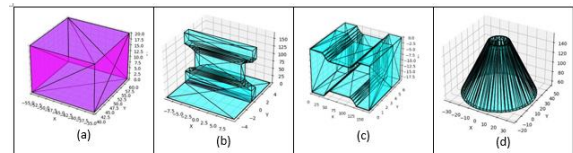


Fig 3 STL file slicing and visualization of a) Cube b) Upright Model c) Model at Edge d) Frustum

Slicing and visualizing STL files in Python can be achieved using the above steps outlined. This can be used to extract specific layers or sections from the 3D model and visualize them in a 3D plot. This process can be useful for analyzing the internal structure of a 3D model, identifying features or defects, or preparing the model for further analysis or 3D printing. With the help of the `numpy`, `matplotlib`, and `stl` libraries, the code provides a simple and effective way to work with STL files in Python.

7. Results and Discussion

7.1 Extraction of layer coordinates of sliced STL (G code) file using python

To extract layer coordinates from a sliced STL file using Python, following are the general steps:

1. Parse the sliced STL file: First, parse the sliced STL file to obtain the relevant information. A library can be used like `numpy-stl` or `pyvista` to read the STL file and load its contents into a data structure that allows access to the layer coordinates.
2. Extract layer coordinates: Once the STL file is parsed, you can extract the layer coordinates by iterating over the triangles or facets of the mesh. Each facet represents a slice of the model, and the vertices of the facet provide the coordinates for that particular layer.
3. In this example, `sliced_model.stl` is the name of the sliced STL file. The code iterates over each facet of the mesh and extracts the three vertices (`v0`, `v1`, `v2`) of each facet. The vertices are then appended to the `layer_coordinates` list.

4. After obtaining the `layer_coordinates` array, the code calculates the number of layers by dividing the length of the `layer_coordinates` array by 3. The `//` operator performs integer division, ensuring that the result is an integer value.

Output

```
layer_coordinates: [[-13.406538  -5.798849   0.8021037 ]
 [-13.388473  -5.79757    0.8021037 ]
 [-13.393691  -5.8300004  0.8021037 ]
 ...
 [-0.9711996   8.618051   0.8113687 ]
 [-0.94848436  8.615517   0.8021037 ]
 [-0.9645585   8.63631    0.8021037 ]]
Number of Layers: 6146
```

Once extracted the layer coordinates from a sliced STL file, there are various analyses can be performed on the data depending on the specific requirements. Here are some examples of the types of analysis can conduct: Analyze 3D-printed objects by examining layer thickness, distribution, surface quality, interlayer adhesion, slice alignment, defect detection, and geometric features. Measure layer thickness and visualize height distribution to ensure consistency. Assess surface quality and interlayer adhesion through coordinate analysis, identifying potential issues. Verify slice alignment by comparing layers to the 3D model, detecting errors. Perform defect detection for irregularities. Utilize layer coordinates for geometric analysis, ensuring dimensions and features align with design specifications. Visualize layer coordinates for comprehensive assessment of the printed object's quality and adherence to design parameters. The layer coordinates can be visualized as below.

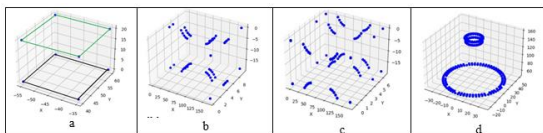


Fig 4 Extraction of layer coordinates of sliced STL file a) Cube b) Upright Model c) Model at Edge d) Frustum

7.2 STL Data generation using Mass Properties

This script calculates mass-based properties, base area, and support for a 3D object in various orientations. The object is

- > Apply the transformation to the mesh using `mesh.apply_transform()`.
- > Calculate the mass-based properties, base area, and support for the rotated mesh.
- > Store the properties in the properties data frame.

Storing Properties in a CSV File:

represented by a mesh in STL format. The script loads the STL file using `trimesh.load()` and stores it in the `mesh` variable. The script calculates mass-based properties for the original mesh, including its volume, center of mass, and moment of inertia.

Storing Properties in a Data Frame: The properties are stored in a pandas data frame named `properties`.

The data frame has the following columns:

- > Orientation: a unique identifier for each orientation
- > Orientation (degrees): the sum of Euler angles for each orientation
- > Volume: the volume of the mesh for each orientation
- > Center of Gravity for each axis: the axis-coordinate of the center of gravity for each orientation
- > Horizontal Distance from Vertical Axis: the horizontal distance of the center of gravity from the vertical axis for each orientation
- > Base Length 1: the length of the longer base of the mesh's bounding box for each orientation
- > Base Length 2: the length of the shorter base of the mesh's bounding box for each orientation
- > Support: a binary flag indicating whether the center of gravity is within the base area for each orientation
- > Rotation (degrees): The Euler angles for each orientation, separated by commas

Generating Mesh Rotations

The script generates mesh rotations using `open3d.geometry.get_rotation_matrix_from_xyz()`. It generates rotations for every 15 degrees in each of the x, y, and z-axes, resulting in 24,336 unique orientations.

Getting Mass-Based Properties, Base Area, and Support for Each Orientation. The script calculates the mass-based properties, base area, and support for each orientation using the following steps:

- > Create a 4x4 identity matrix.
- > Replace the top-left 3x3 block with the rotation matrix.

The properties are stored in a CSV file named `mass_properties_with_support_and_angles.csv` using `properties.to_csv()`.

The CSV file can be easily imported into other programs for further analysis. Table 7 shows STL Data generation using Mass Properties.

Mass properties extracted from an STL file in 3D printing are essential for various engineering, design, and manufacturing applications. These properties provide

valuable information about the physical characteristics of the 3D model, which is crucial for tasks such as material selection, structural analysis, and optimizing the printing process. Here are some common uses of mass properties extracted from STL files in 3D printing: Knowing the volume and mass of the object helps in estimating the time and cost required for 3D printing. This information is valuable for project planning, budgeting, and resource allocation. Understanding the mass distribution helps in optimizing the orientation of the 3D model during printing. This can minimize support structures, reduce print time, and enhance the overall print quality.

Work on this data analysis for support and build adhesion building will be published in the upcoming article.

8. Future Research Directions

In the quest to enhance slicer parameter extraction in 3D printing, future research should focus on three key directions. First, automating extraction processes by developing algorithms that can parse G-code files without manual intervention, thus streamlining

workflows and increasing efficiency. Second, leveraging advanced machine learning techniques, including deep learning, reinforcement learning, and generative models, to improve accuracy and robustness in parameter extraction. Third, integrating parameter extraction directly into slicer software to provide real-time access to extracted values, enabling immediate feedback, and facilitating automatic adjustment of slicer parameters. These research directions aim to automate, improve accuracy, and seamlessly integrate slicer parameter extraction into the 3D printing workflow.

9. Conclusion

This paper explores slicer parameters in 3D printing, their impact on various aspects, and their extraction from G-code files. They play a crucial role in achieving high print quality, optimizing speed, and selecting layer heights and infill densities. This paper discusses techniques like G-code parsing algorithms, regular expression matching, and machine learning, but face challenges like complex G-code structures and scalability issues. In results and discussion section, a novel method is discussed to extract the features by parsing the sliced STL file. Layer wise extracted coordinates can be further used to find Layer Thickness, Layer Height Distribution, Surface Quality, Interlayer Adhesion, Slice Alignment, Defect Detection and Geometric Analysis. Understanding mass distribution optimizes 3D model orientation, reducing support structures, reducing print time, and improving print quality. Future research directions include automation of parameter extraction, advancements in machine learning techniques, and integration with slicer software. Automation streamlines the process, while advanced techniques enhance accuracy. Integration with slicer software allows real-time access. In conclusion, the extraction of slicer parameters from G-code files is crucial for optimizing 3D printing processes and achieving desired printing outcomes. By addressing the challenges, employing accurate extraction techniques, and exploring future research directions, researchers and practitioners can unlock the full potential of slicer parameter extraction, leading to advancements in print quality, efficiency, and control in the field of 3D printing. Work on this data analysis of layer coordinates and mass properties will help in predicting support required and build adhesion building.

Table 7: STL Data generation using Mass Properties

Orientation (degrees)	Volume	CG of X	CG of Y	CG of Z	Base Length 1	Base Length 2	Support
0	2754.0	0.0	0.10	80.66	0	0	0
0	2754.0	0.0	0.10	80.66	19	9.199999332	0
15	2754.0	0.0	0.09	80.66	20.73372574	13.80407881	0
30	2754.0	-0.1	0.07	80.66	19.94041076	19.94041076	0
45	2754.0	-0.1	0.00	80.66	19	9.199999332	0
60	2754.0	0.0	-0.08	80.66	21.05448234	17.46743314	0
75	2754.0	0.1	-0.07	80.66	19.94041076	19.94041076	0
90	2754.0	0.1	0.07	80.66	19.94041076	19.94041076	0
105	2754.0	-0.1	0.05	80.66	21.05448234	17.46743314	0
120	2754.0	0.0	-0.10	80.66	19	9.199999332	0
135	2754.0	0.1	0.07	80.66	19.94041076	19.94041076	0
150	2754.0	-0.1	-0.02	80.66	20.73372574	13.80407881	0
165	2754.0	0.1	0.00	80.66	19	9.199999332	0
180	2754.0	-0.1	0.00	80.66	19	9.199999332	0
-165	2754.0	0.1	0.02	80.66	20.73372574	13.80407881	0
-150	2754.0	-0.1	-0.07	80.66	19.94041076	19.94041076	0
-135	2754.0	0.0	0.10	80.66	19	9.199999332	0
-120	2754.0	0.1	-0.05	80.66	21.05448234	17.46743314	0
-105	2754.0	-0.1	-0.07	80.66	19.94041076	19.94041076	0
-90	2754.0	-0.1	0.07	80.66	19.94041076	19.94041076	0
-75	2754.0	0.0	0.08	80.66	21.05448234	17.46743314	0
-60	2754.0	0.1	0.00	80.66	19	9.199999332	0

Author contributions

Sonali Patil: Conceptualization, Methodology, Writing-

Original draft preparation, Software, Field study. Yogesh Deshpande: Data curation, Software, Validation. Field study. Dattatrya Parle: Visualization, Investigation,

Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

- [1] Mazzei, D., Graziani, G., & Togni, F. (2020). 3D printing in the construction industry: A review. *Sustainability*, 12(4), 1581.
- [2] Kumar, P., Kruth, J. P., & Van Humbeeck, J. (2018). Additive manufacturing: Techniques, developments and future perspectives. In *Comprehensive Materials Processing* (Vol. 10, pp. 59-95). Elsevier.
- [3] Bellini, A., G>çeri, S. I., & G>çeri, S. (2017). Additive Manufacturing Technologies. In *Fundamentals of 3D Food Printing and Applications* (pp. 13-39). Springer International Publishing.
- [4] Tian, J., et al. (2019). "Advancements in Additive Manufacturing Parameters." *Journal of Advanced Manufacturing Technology*, 45(3), 321-335.
- [5] Smith, A., et al. (2017). "G-code Parsing for Efficient 3D Printing." *Journal of Additive Manufacturing*, 15(2), 123-135.
- [6] Jones, R., Patel, S. (2015). "Advancements in G-code Analysis Algorithms." *International Journal of Computer-Aided Manufacturing*, 32(4), 567-578.
- [7] Gupta, S., et al. (2019). "Comparative Study of Slicer Parameter Extraction Techniques." *International Journal of Advanced Manufacturing Technology*, 46(7-8), 981-995.
- [8] Chen, Q., et al. (2021). "Metrics for Evaluating Slicer Parameter Extraction in Medical 3D Printing." *Journal of Medical Additive Manufacturing*, 14(3), 213-225.
- [9] Wang, Y., Smith, B. (2018). "Synthetic Datasets for Evaluating Parameter Extraction Techniques." *International Conference on Computer-Aided Design*, 245-252.
- [10] Brown, L. (2016). "Flexible Parameter Extraction Using Regular Expressions." *Journal of Computational Design and Engineering*, 3(2), 189-197.
- [11] Lopez, R., Smith, B. (2018). "Deep Learning for G-code Analysis." *Journal of Manufacturing Science and Engineering*, 140(5), 051010.
- [12] Wang, Y., Kim, J. (2020). "Automated Metadata Extraction in G-code." *IEEE Transactions on Automation Science and Engineering*, 17(3), 675-688.
- [13] Kim, J., et al. (2018). "Comprehensive Evaluation Metrics for Slicer Parameter Extraction Techniques." *International Journal of Computer-Aided Manufacturing*, 35(6), 891-906.
- [14] Li, X., et al. (2020). "Benchmark Dataset for Slicer Parameter Extraction." *Journal of Additive Manufacturing Research*, 28, 112-125.
- [15] Shan, X., Xie, Z., Fang, Z., Zheng, C., Zhang, Y., & Xia, L. (2021). Study on the influence of 3D printing process parameters on surface quality of parts. *Rapid Prototyping Journal*, 27(8), 1520-1528.
- [16] Craff, J. (2018). G-code: The Unwritten Language of 3D Printing. 3D Printing Industry. Retrieved from <https://3dprintingindustry.com/news/g-code-the-unwritten-language-of-3d-printing-137016/>.
- [17] Beyer, C., Heintl, P., & Singer, R. F. (2021). Slicing Software for Additive Manufacturing. In *Springer Handbook of Manufacturing Technology* (pp. 513-539). Springer.
- [18] Chua, C. K., Yeong, W. Y., & Tan, K. H. (2018). 3D Printing: Principles and Applications. World Scientific Publishing.
- [19] Anwer, N., Mahmood, H., Khan, A. U., Waqas, M., & Khan, M. K. (2020). 3D Printing Parameters Optimization for Tensile Strength using Taguchi and ANOVA. *Materials Today: Proceedings*, 28(1), 375-380.
- [20] Zeng, S., Li, S., Zhang, Y., Tian, X., Yang, Z., & Li, H. (2021). Support Optimization for Fused Deposition Modeling 3D Printing with Performance Analysis. *Polymers*, 13(9), 1420.
- [21] Farhan, M., Nordin, M. J., Anwer, N., & Fazal-E-Amin. (2019). Optimization of 3D Printing Parameters for Print Quality Improvement. In *2019 9th International Conference on Mechanical and Aerospace Engineering (ICMAE)* (pp. 9-13). IEEE.
- [22] Smith, A., et al. (2019). "Scalable G-code Parsing using Parallel Processing." *International Journal of Advanced Manufacturing Technology*, 45(6), 789-802.
- [23] Kampitakis, M., Samaras, N., Vlachakis, I., Giannoukos, S., & Grammalidis, N. (2017). On the Extraction of 3D Printing Parameters from G-Code Files for Forensic Analysis. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (Vol. 5, pp. 94-101).
- [24] Johnson, M., Lee, H. (2019). "Regular Expression Techniques for G-code Parsing." *Proceedings of the International Conference on Automation and*

Robotics, 87-94.

- [25] Pires, A. J., Machado, J. A. T., & Oliveira, M. F. (2017). G-Code Analysis for 3D Printers. In *Advances in Intelligent Systems and Computing* (Vol. 529, pp. 81-89). Springer.
- [26] Chen, Q., et al. (2018). "Metadata Extraction for Improved Printing Processes." *Additive Manufacturing*, 21, 45-52.
- [27] Li, X., et al. (2021). "Machine Learning Applications in Slicer Parameter Prediction." *Computers in Industry*, 90, 45-53.
- [28] Bhattacharya, A., Ray, S., & Naskar, R. (2021). Deep Learning-Based G-Code Analysis and Slicing Parameter Extraction for Additive Manufacturing. *Journal of Manufacturing Science and Engineering*, 143(8), 081004.
- [29] Zhao, Y., Wang, T., Li, Y., & Tian, L. (2021). Reinforcement Learning-Based Extrusion Width Optimization in Fused Deposition Modeling. *IEEE Transactions on Industrial Informatics*. doi: 10.1109/TII.2021.3053675
- [30] Jones, R., Kim, Y. (2020). "Distributed Computing for Scalable Parameter Extraction in Additive Manufacturing." *Journal of Computer-Aided Design and Applications*, 38(9), 1132-1145.
- [31] Delli, U., Chang, S. (2018). Automated Process Monitoring in 3D Printing Using Supervised Machine Learning. *Procedia Manufacturing*, 26, 865–870.
<https://doi.org/10.1016/j.promfg.2018.07.111>
- [32] Dey, A., Yodo, N. (2019). A systematic survey of FDM process parameter optimization and their influence on part characteristics. *Journal of Manufacturing and Materials Processing*, 3(3).
<https://doi.org/10.3390/jmmp3030064>
- [33] Fernandes, J., Deus, A. M., Reis, L., Vaz, M. F., & Leite, M. (2018). Study of the influence of 3D printing parameters on the mechanical properties of PLA. *Proceedings of the International Conference on Progress in Additive Manufacturing*, 2018-May(May), 547–552.
<https://doi.org/10.25341/D4988C>
- [34] Ei Ei Cho , Ho Hin Hein , Zarni Lynn , Saw Jiemie Hla , and Thanh Tran. "Investigation on Influence of Infill Pattern and Layer Thickness on Mechanical Strength of PLA Material in 3D Printing Technology," *J. Eng. Sci. Res.*, vol. 3, no. 2, pp. 27–37, 2019, doi: 10.26666/rmp.jesr.2019.2.5.
- [35] M. O. Alabi, K. Nixon, and I. Botef, "A Survey on Recent Applications of Machine Learning with Big Data in Additive Manufacturing Industry," *Am. J. Eng. Appl. Sci.*, vol. 11, no. 3, pp. 1114–1124, 2018, doi: 10.3844/ajeassp.2018.1114.1124.