Verification of Data Compression Focusing on Continuity in 3D Printing

Satoshi Kodama
Department of Information Technology, Faculty of Technology
International Professional University of Technology in Tokyo, Japan
kodama.satoshi@iput.ac.jp

Abstract Recently, 3D printers have become capable of producing relatively large, high-resolution models. Unlike simple shapes, it is becoming possible to print large complex shapes with high accuracy. However, the data size of complex models is also large, and the slice data required for printing is also large. Thus, in this study, we investigated reducing the data size by focusing on the characteristics of the slice data required for 3D printing. The proposed method focuses on the continuity of each layer and the top/bottom layers of the cross-section used to print the 3D model. Preliminary experiments were conducted to determine whether the data size could be reduced by applying the difference method. Here, the results obtained from the continuity were output as text data, and various metadata, e.g., lamination pitch data, required for printing were ZIP compressed. Then, we compared conventional file formats as a format that can be converted as a printable file as lossless compression. The results demonstrated that the file size can be reduced for 3D complex shapes with a large number of vertices, which are difficult to handle. We found that the proposed difference method was effective for relatively large files that require a general-purpose graphics processing unit to create slice data.

Keywords: 3D printing, data compression, stereolithography, additive manufacturing, sliced data, image storage

1. Introduction

With the ongoing development of three-dimensional (3D) printers, it has become relatively easy to output high-definition, large-scale fabrication objects with complex shapes, which is difficult to do using conventional fabrication methods. Currently, 3D printers are used in many fields [2, 44, 52]. However, the need to handle a wide range of high-definition data has resulted in increased data sizes and the emergence of various file formats depending on the printer used for output, thereby increasing the difficulty of data handling. In addition, when creating cross-sectional views of each layer required for printing from 3D model data, the load for analysis increases [30, 34, 58, 59].

Generally, when printing objects with a 3D printer, the 3D data are decomposed into two-dimensional (2D) data for each layer, and then laminated based on the stacked 2D data [30, 31, 49, 56]. In this study, we conducted an initial verification to determine whether it is possible to reduce the data size using lossless compression based on the continuity of the sliced data required for 3D printing output. As a result, although this was preliminary experiment, we found that combining the proposed method and ZIP compression resulted in a data size that was smaller than that of the conventional large

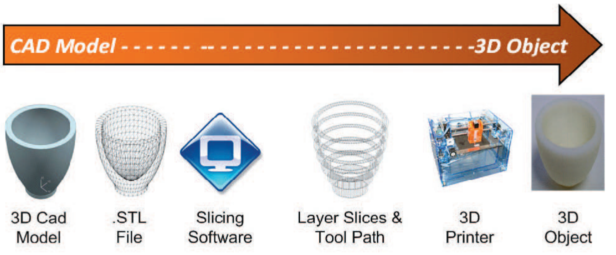


Fig. 1. The 3D Printing process (from [56], license: CC BY 3.0).

data format. In particular, the effectiveness of the proposed method was demonstrated for geometries with a large number of vertices, which tend to be complex geometries. On the other hand, we could not demonstrate the usefulness of the proposed method for 3D models with a small number of vertices. Therefore, the proposed method is effective for compression of 3D model data that requires time for shape analysis necessary for printing. In the future, based on this result, it is thought that even higher compression will be possible by considering such as the symmetry of slice data.

2. Related Work

Printing high-definition, relatively large-sized objects has become easier; thus, 3D printers are widely used in a variety of fields [2, 44, 52]. In addition, 3D printers with various output formats are currently available from multiple manufacturers, and even inexpensive consumer grade printers are increasing in terms of both size and print definition [1, 17].

However, it is becoming increasingly difficult to handle the various manufacturer-specific data formats and the large amounts of data required for 3D printing [3, 12, 21, 24, 38, 46].

Therefore, in this section, we introduce the main 3D printing methods and the methods used to create the data required for printing and file formats. In addition, we identify known problems with these existing methods.

2.1. Types of 3D printers

Various output methods are being researched and developed for 3D printers; however, the most common method is to transform a 3D model as a layer of 2D data and then laminate these layers in sequence (Fig. 1 [56]). Here, we introduce two common lamination methods, e.g., the fused deposition modeling (FDM) and resin printing technology methods.

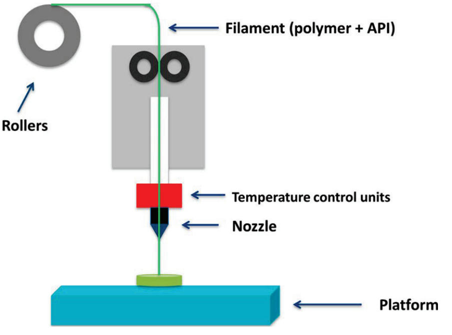


Fig. 2. FDM printing method (from [32], license: CC BY 4.0).

2.1.1. Production of 3D objects by heat

Generally, the well-known FDM method realizes 3D printing by melting a filament in a solid state at a high temperature and stacking it layer-by-layer through a nozzle (Fig. 2) [15, 32]. The FDM method is commonly used, including by individual users, and it has become increasingly affordable since the patent expired in 2009 [35, 50]. Currently, filaments of various materials are readily available, and it is possible to change the hardness and the like according to the use purpose [15]. However, although the structure is very simple, this method is susceptible to various output errors, e.g., heat shrinkage and prints detaching from the build platform, depending on the filament material and the ambient temperature of the print environment [42].

2.1.2. Production of 3D objects by ultraviolet light

Here, ultraviolet-curable resin is cured in a layer-by-layer manner by surface irradiation of ultraviolet light to produce a modeled object. 3D printing using resin primarily includes stereolithography apparatus (SLA), digital light processing (DLP), and liquid crystal display (LCD) depending on the ultraviolet irradiation method (Fig. 3) [25]. With the SLA method, an ultraviolet laser is applied from the bottom or top of a tank filled with photocurable resin for curing. The SLA involves tracing and curing a layer based on the 2D image as a point, and after creating a surface, the surface of the next layer is created in the same way (Fig. 4a) [53]. The DLP method replaces the laser with a projector and applies ultraviolet rays as a digital image. Unlike lasers, the DLP method can irradiate a wide range of ultraviolet rays simultaneously; thus, high-speed output is possible (Fig. 4b) [53]. The LCD method cures using an ultraviolet light as the backlight of an LCD panel, and this method can handle more precise modeling by using a high-resolution panel [19, 37].

A primary characteristic of these methods is that they require cleaning and secondary curing as postprocessing due to the resin characteristics; however, they can achieve a

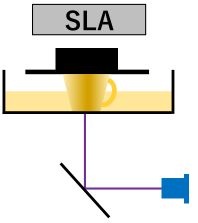
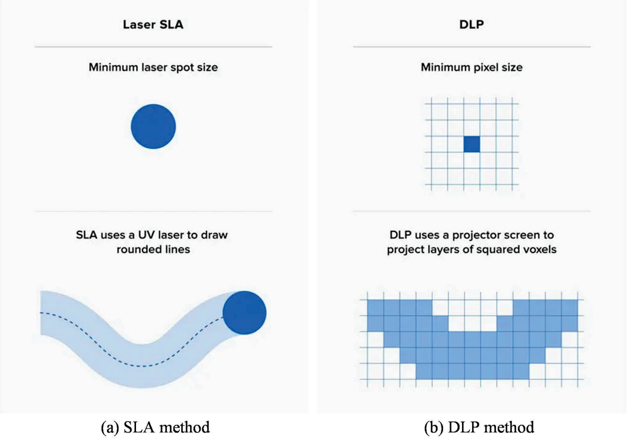


Fig. 3. FDM printing method (created by the author on the basis of [25]).



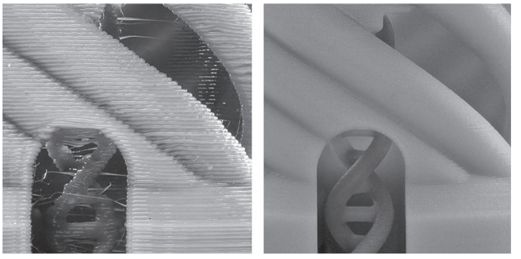
(a) SLA method (b) DLP method

Fig. 4. Differences between (a) SLA and (b) DLC curing methods (from [53], license: CC BY 4.0).

cleaner finish than the FDM method without leaving stacking marks (Fig. 5) [13]. Note that the LCD method is frequently used due to its ability to print at high speed from surface emitting and low cost due to its simple structure. In addition, the LCD system has advanced to higher resolutions, and affordable consumer grade 4K and 8K devices are available.

2.2. Generating printing data

Generally, in the 3D printing process, a cross-sectional view is first created from the model data using slicer software [20, 34, 59]. In the following, we describe the 3D data file format required for printing and how to create 2D data using slicer software.



a **b**

Fig. 5. Differences in output results between (**a**) FDM and (**b**) SLA methods (from [13]; available also from a number of other blogs).

$$
\begin{array}{l}
\textbf{solid } \textit{name} \\
\left\{ \begin{array}{l}
\textbf{facet normal } n_i n_j n_k \\
\quad \textbf{outer loop} \\
\quad\quad \textbf{vertex } v1_x \ v1_y \ v1_z \\
\quad\quad \textbf{vertex } v2_x \ v2_y \ v2_z \\
\quad\quad \textbf{vertex } v3_x \ v3_y \ v3_z \\
\quad \textbf{endloop} \\
\textbf{endfacet}
\end{array} \right\}^{+} \\
\textbf{endsolid } \textit{name}
\end{array}
$$

Fig. 6. STL file specification (created by the author on the basis of [16]).

2.2.1. Production of 3D objects by ultraviolet light

To create a 3D object, it is first necessary to model the 3D data. Various software tools are available for 3D modeling, from freely available software to high-performance commercial software. In addition, there are various file formats for the 3D data; however, the STL and OBJ formats are widely used because they are highly versatile [11, 40].

The STL and OBJ formats are based on vector information, e.g., vertex information; thus, the data size is not related to the size of the object (Fig. 6). However, the amount of data increases for precise shapes, which inevitably leads to increased data sizes [29,30]. In other words, due to the high-definition of 3D printers, their use has increased, and the data size tends to increase even with this format, which is vector information [27].

The size of the data to be printed varies greatly depending on the size and lamination pitch of the production (Fig. 7). For example, if the stacking width is 100 µm, which is common, 1000 stacks are required to print a 10 cm object, which means that 1000 cross-sectional views must be generated. In recent years, it has become possible to achieve a lamination process of 25 µm or less, which inevitably increases the amount of data [26].

2.2.2. Creating slice data for printing

As mentioned previously, 3D printers create 3D objects by stacking layers; thus, it is necessary to create cross-sectional views corresponding to each layer from the model data

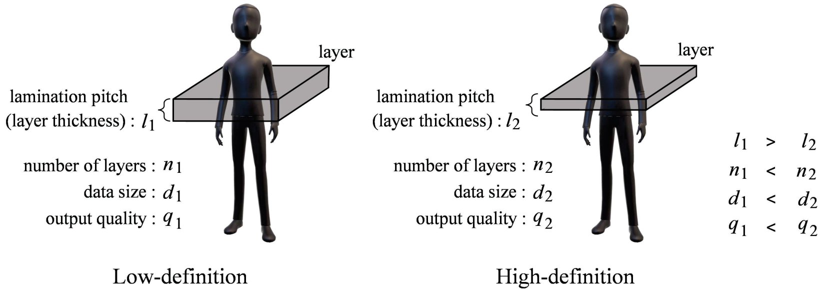


Fig. 7. Lamination pitch.

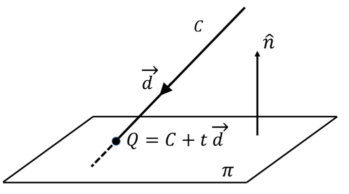


Fig. 8. Intersection between a straight line and a plane (from [23], license: CC BY 4.0).

(Fig. 1) [34,56,59]. Generally, slicer software is used to create a cross-sectional view that is compatible with a given 3D printer; however, some modeling software has a slicing function that creates a cross-sectional view [9].

Various methods can be used to create a cross section for 3D printing; however, the most common method is to use the intersection of a plane and a straight line (Fig. 8) [23]. In addition, the slicing algorithm using the intercept theorem for triangular mesh location information has also been studied [51]. Furthermore, a similar study is developing slice algorithm that utilizes the geometric topology information of the STEP model conforming to ISO10303 [54].

However, it has been demonstrated that the process of creating cross sections is generally time consuming. Recently, as the resolution has increased, converting 3D data to 2D data incurs an extremely long processing time; thus, methods that use a general-purpose graphics processing unit (GPGPU) are being actively investigated [20,34,59].

2.2.3. Slice data file format

The model data are converted into multiple pieces of 2D image data by the slicer software; however, when outputting, it is necessary to select a file format that is suitable for the given 3D printer. Common slicer software used for conversion, e.g., CHITUBOX, supports a wide range of formats, including PWS and CWS, which are used by several

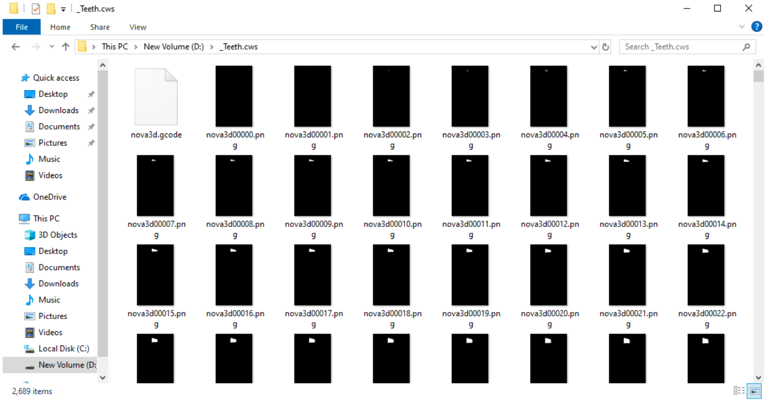


Fig. 9. CWS file format in which the image file and print settings file are ZIP compressed.

common 3D printers, e.g., ELEGOO, Anycube, and Flashforge printers, as well as the more generic SLC [12,43]. There is also a general-purpose storage method in which sliced images are compressed in ZIP format with various information included, e.g., lamination pitch [4]. Here, the 2D data can be saved as a raster format image, e.g., CWS format (Fig. 9), or saved by internal and external boundary polylines, e.g., SLC files [10].

Generally, when printing a 3D object from multiple images, specific information is required, e.g., lamination pitch; thus, a metadata file containing the information required for printing is frequently added (Fig 10). However, some devices can print 3D objects directly from multiple 2D image data by setting them at the time of printing [41, 57].

2.3. Template Matching

Template Matching is one of the famous and fundamental Computer Vision technique for Object Detection/Recognition. Via Template Matching, the computer figures out whether the test image contains a given template in it or not [36]. In most situations the sum of absolute differences (SAD) and Sum of squared differences (SSD) are used as similarity measures to find the best similar block [48].

2.3.1. Pixel Differencing method using SAD

SAD is a technique for evaluating the similarity between two same size regions, and widely used in stereovision, optical flow, motion estimation and so on [45]. The total difference between the two signatures is calculated by adding the absolute value of differences between the samples. The match with the smallest total difference is taken as

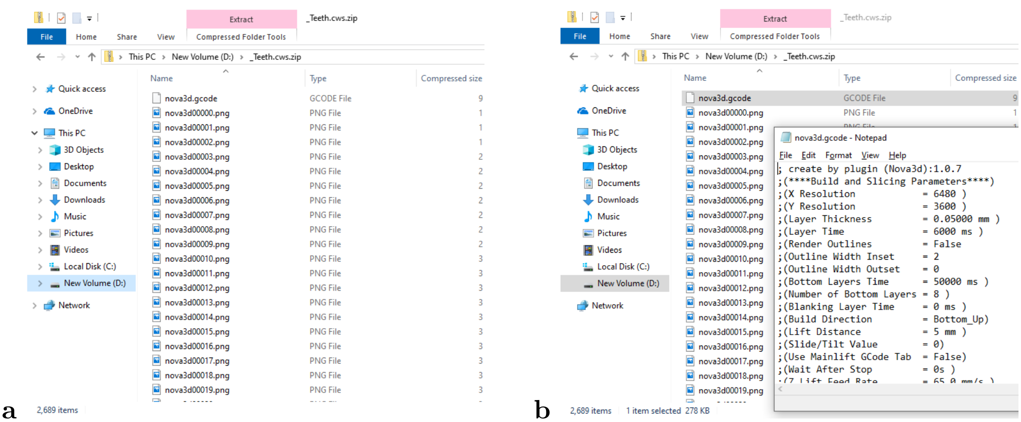


Fig. 10. Content of files required for printing (created by the slicer from model images). (**a**) Image data and metadata in print files. (**b**) Content of the metadata generated by the slicer.

best [14]. This method can be expressed as follows:

$$S_{\mathrm{SAD}}(x,y) = \sum_{x=1}^{M}\sum_{y=1}^{N}\Big|T(x,y) - I(x+u,y+v)\Big|,$$

where M is size of rows in reference image and N is size of column while u and v are variable, shift component along x-direction and y-direction, respectively. $T(\cdot,\cdot)$ and $I(\cdot,\cdot)$ represent the value of the pixel at a location. If the images exactly match, the result will be zero [33].

2.3.2. Pixel differencing method using SSD

SSD is one of measures of match that is based on pixel by pixel intensity differences between the two images [47]. It calculates the summation of squared for the product of pixels subtraction between two images [55]. This method can be expressed as follows:

$$S_{\mathrm{SSD}}(x,y) = \sum_{x=1}^{M}\sum_{y=1}^{N}\left(T(x,y) - I(x+u,y+v)\right)^2 .$$

The SSD score, like the SAD score, must be zero if the images are the same, pixel by pixel.

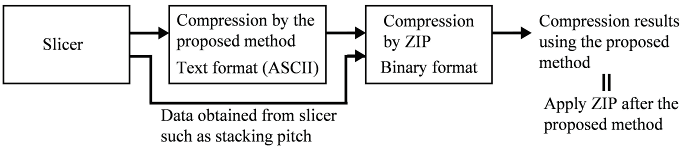


Fig. 11. Process flow of the proposed method.

3. Proposed Method

In this study, as an initial verification, we perform a data compression method that focuses on continuous data unique to 3D printers. Specifically, we target stereolithography, which can high-resolution products, and we attempted to perform lossless compression that can be converted into printable data.

Generally, the data printed by a 3D printer are characterized by relatively similar upper and lower layers. In addition, the objects floating in the air cannot be directly printed by a 3D printer because they must be suspended from another object that serves as a base. Therefore, due to these characteristics, as an initial experiment, we verify whether it is possible to reduce the data size of the 3D model required for printing.

Here, in consideration of versatility, the input data were 2D slice image data and the metadata required for printing obtained from a slicer. Thus, the slice data and metadata were converted from the 3D model by the slicer in advance. This was a preliminary experiment; thus, we output the intermediate results as text data to facilitate verification of the proposed method. Finally, the results of the proposed method, which were compressed in ZIP format including metadata, were compared with other file formats (Fig. 11).

3.1. Creating difference data from sliced data

The slicer converted the 3D data into 2D data for each layer, and scanned the generated 2D data for each XY coordinate (Fig. 12). Here, the scanned image data contained binary information; thus, it was possible to easily obtain internal and external information scanned (Fig. 13). In other words, it was possible to identify positions that change from outside to inside or inside to outside; thus, the differences in positions were stored sequentially. In this verification, the difference of each X coordinate was obtained in order along the y-axis (Fig. 14), and the same process was performed for all layers to obtain the difference data at each layer.

As mentioned previously, objects fabricated using a 3D printer frequently have approximate top and bottom layers, and it is impossible to print objects that are suspended in air that do not touch the print surface or are not connected to the object being printed.

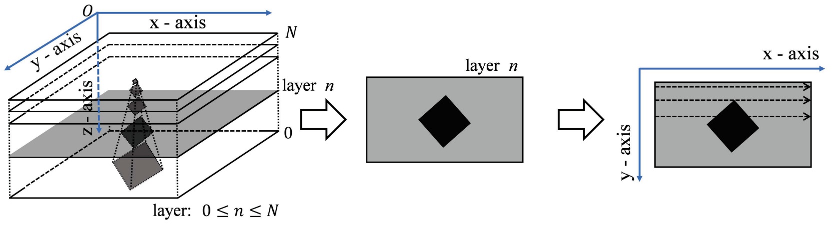


Fig. 12. Scanning method for internal and external determination.

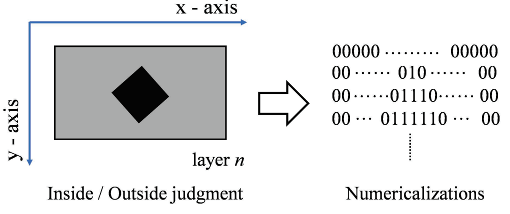


Fig. 13. Judgment by binarization.

Thus, some data will be continuous even in the upper and lower layers. Therefore, after obtaining the difference data in each layer, the difference based on the Z-axis was considered. Note, as this study is an initial validation, it was decided to use the index of the data in the upper layer if it is equal to the upper layer (Fig. 15 and Fig. 16). In other words, this is the case of perfect matching in SSD, and this method is an extended solution.

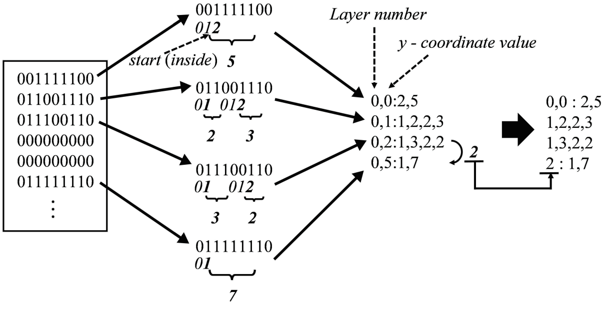


Fig. 14. Numericalization of continuous data for each layer.

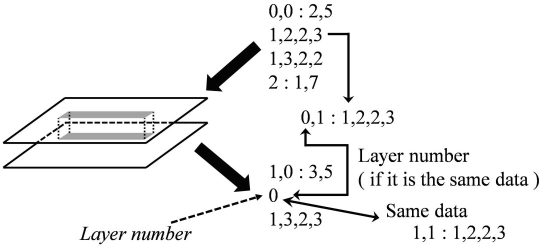


Fig. 15. If the y-axis values are the same data.

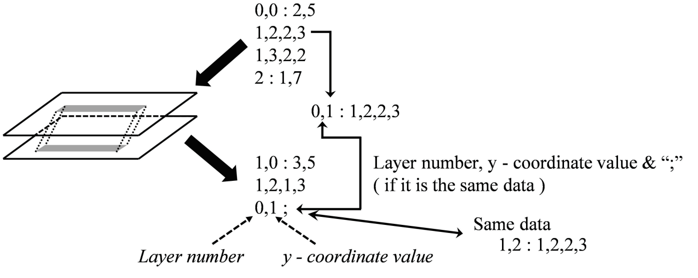


Fig. 16. If the y-axis values are different data.

3.2. Compression of numerical data by proposed method including metadata

For the final comparison, the text data created by the proposed method and other data, e.g., laminate pitch data obtained by the slicer, were combined and compressed together in ZIP format, and compared with each file format (Fig. 11). Note that this propose method utilizes lossless compression; thus, it can be changed to a format that can be printed with a 3D printer, e.g., the CWS format.

4. Experiments

Based on the proposed method, we verified whether there any difference could be observed in the data compression results compared with other formats. Here, the experiments were conducted assuming a large print size (298 mm × 164 mm × 30 mm) that can be output by a conventional optical 3D printer. Thus, the size of each experimental 3D model was increasd or decreased to the maximum size within the above range. In addition, the lamination pitch was set to 50 µm. Note that the size of the STL and OBJ files used in the experiment could be changed easily because the data are based on coordinate information.

Tab. 1. Specifications of experimental hardware and software.

Specification	Value
OS	Windows 11 Professional (Build 22621)
Compiler	Javac 19.0.1 (Oracle JDK 19)
Slicer	CHITUBOX V1.8.0 Beta
	Nova3D Plugin 1.07
	anycubic_plugin
CPU	Intel Core i9 - 12900H (2.5 GHz)
GPU	GeForce RTX 3070 Ti Laptop
Memory	64 GB (DDR5)

As mentioned earlier, printing requires a sliced 2D image of each layer. Generally, there are several software programs that convert such 3D data into 2D images, but this time, we decided to use CHITUBOX, which is freely available and compatible with many 3D printers [12]. CHITUBOX, like a general slicer, also generates configuration files necessary for printing. Configuration files other than images created with the CHITUBOX are also saved together with the data in the proposed method during ZIP compression. Therefore, in the evaluation, we will compare proposed file format with other file formats as printable files. The file formats to be compared were OBJ files or STL files that are model data, and the formats used by general stereolithography 3D printers Anycube, Phrozen, NOVA3D, and general-purpose SLC formats.

In this experiment, compression was applied and validated on eight shapes. The experimental hardware and software are described in Tab. 1. In the following results, the results of the proposed method are referred to as apply ZIP after the proposed method. As mentioned earlier, the result of the ZIPping process applied after the proposed method can be easily decompressed and converted to a printable file because it contains the control file for the 3D printer.

4.1. King kong bust

The complex shape model created in OBJ format shown in Fig. 17 was used for verification, and a portion of the cross-sectional view of Fig. 17 is shown in Fig. 18. The difference in data size for each file format is shown in Tab. 2. Note that this model contained 7,560,074 vertices and 2,985 layers.

4.2. Triceratops

The complex shape model created in STL format shown in Fig. 19 was used for verification, and a portion of the cross-sectional view of Fig. 19 is shown in Fig. 20. The

Fig. 17. Complex King kong bust 3D model [28] (Royalty Free No Ai License from “cgtrader | General Terms and Conditions | 21A. Royalty Free License”).

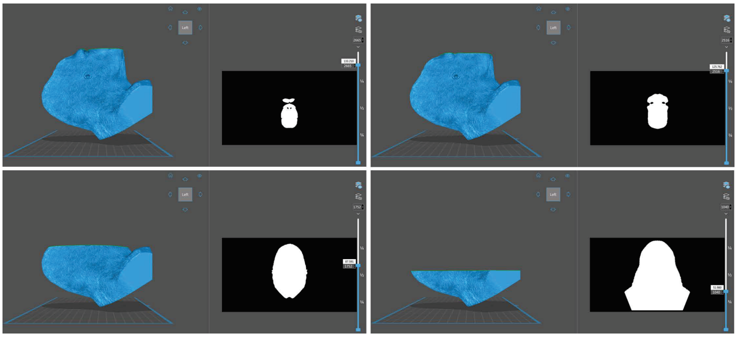


Fig. 18. Part of the cross-sectional view of the King kong bust 3D model.

difference in data size for each file format is shown in Tab. 3. Note that this model contained 1 278 259 vertices and 2 198 layers.

Tab. 2. Data size of various file formats and data size after compression by the proposed method (King kong bust: 7 560 074 vertices and 2 985 layers).

File type	File size [bytes]
OBJ (original) [28]	529 351 777
OBJ (after ZIP compression)	158 387 269
PWS	562 164 661
Photon	566 937 601
ZIP (2D image layers + metadata)	60 117 977
slc	204 691 406
phz	611 932 743
cws	56 439 318
pwx	56 940 340
svgx	381 507 904
Apply ZIP after the proposed method	14 730 722

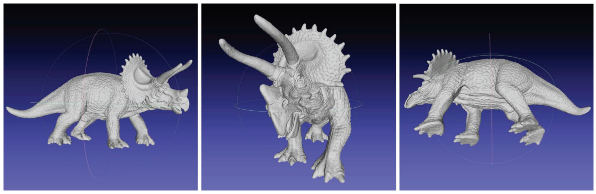


Fig. 19. Complex triceratops 3D model [18] (Royalty Free No Ai License from "cgtrader | General Terms and Conditions | 21A. Royalty Free License").

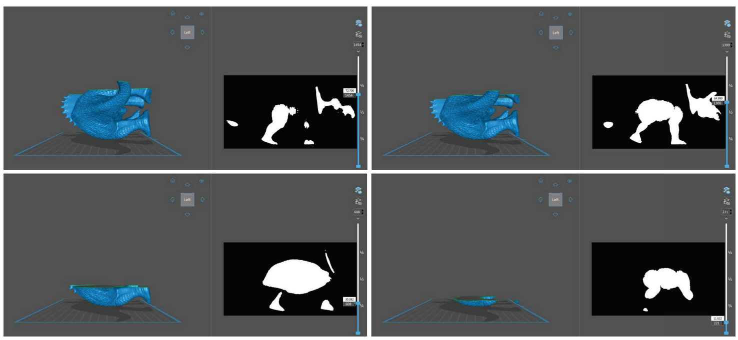


Fig. 20. Part of the cross-sectional view of the triceratops 3D model.

Tab. 3. Data size of various file formats and data size after compression by the proposed method (Triceratops: 1 278 259 vertices and 2 198 layers).

File type	File size [bytes]
STL (original) [18]	127 829 284
STL (after ZIP compression)	54 745 386
PWS	414 868 236
Photon	418 408 591
ZIP (2D image layers + metadata)	54 211 661
slc	69 585 774
phz	453 295 532
cws	51 863 803
pwx	46 609 868
svgx	129 712 549
Apply ZIP after the proposed method	20 520 640

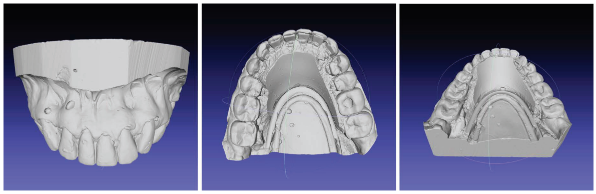


Fig. 21. Complex Plaster cast of teeth 3D model [7] (license: CC BY 4.0, by Artec 3D).

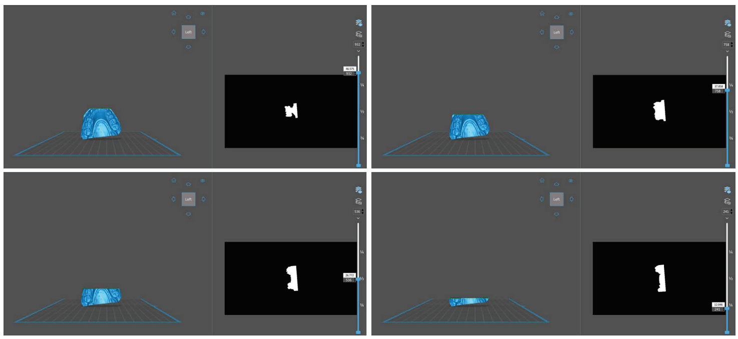


Fig. 22. Part of the cross-sectional view of the Plaster cast of teeth 3D model.

4.3. Plaster cast of teeth

The complex shape model created in OBJ format shown in Fig. 21 was used for verification, and a portion of the cross-sectional view of Fig. 21 is shown in Fig. 22. The difference in data size for each file format is shown in Tab. 4. Note that this model contained 999 998 vertices and 2 688 layers.

4.4. Motorcycle engine HD

The complex shape model created in OBJ format shown in Fig. 23 was used for verification, and a portion of the cross-sectional view of Fig. 23 is shown in Fig. 24. The difference in data size for each file format is shown in Tab. 5. Note that this model contained 999 808 vertices and 3 422 layers.

4.5. Giraffe skull

The complex shape model created in OBJ format shown in Fig. 25 was used for verification, and a portion of the cross-sectional view of Fig. 25 is shown in Fig. 26. The difference in data size for each file format is shown in Tab. 6. Note that this model contained 744 647 vertices and 2 540 layers.

Tab. 4. Data size of various file formats and data size after compression by the proposed method (Plaster cast of teeth: 999 998 vertices and 2 688 layers).

File type	File size [bytes]
OBJ (original) [7]	81 400 987
OBJ (after ZIP compression)	23 484 543
PWS	505 540 271
Photon	509 593 239
ZIP (2D image layers + metadata)	38 397 319
slc	66 177 342
phz	550 203 080
cws	34 151 966
pwx	52 675 512
svgx	120 725 818
Apply ZIP after the proposed method	7 493 959

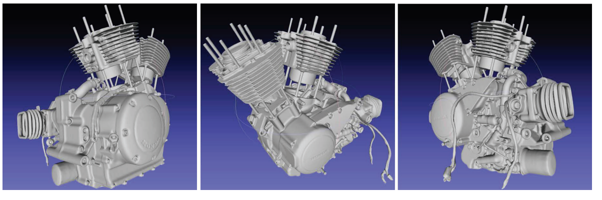


Fig. 23. Complex Motorcycle engine HD 3D model [6] (license: CC BY 4.0, by Artec 3D).

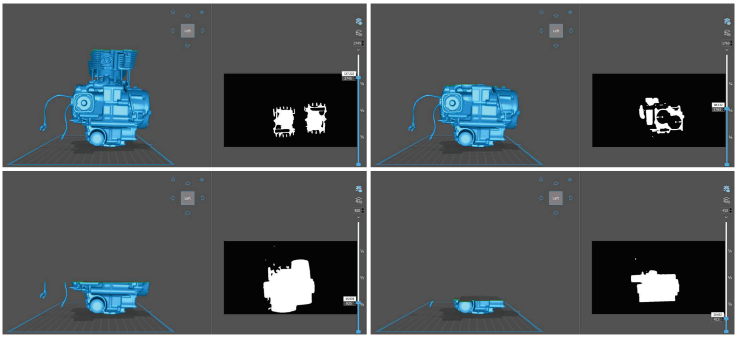


Fig. 24. Part of the cross-sectional view of the motorcycle engine HD 3D model.

Tab. 5. Data size of various file formats and data size after compression by the proposed method (Motorcycle engine HD: 999 808 vertices and 3 422 layers).

File type	File size [bytes]
midrule OBJ (original) [6]	83 283 059
OBJ (after ZIP compression)	30 086 007
PWS	648 374 184
Photon	653 719 793
ZIP (2D image layers + metadata)	63 941 331
slc	89 001 582
phz	712 843 758
cws	62 687 086
pwx	85 320 032
svgx	164 447 945
Apply ZIP after the proposed method	26 801 339

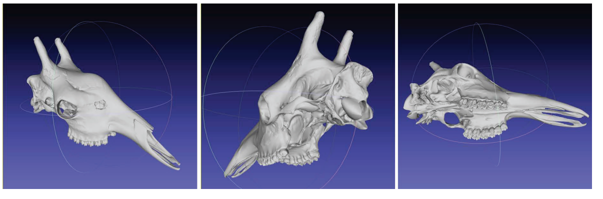


Fig. 25. Complex giraffe skull 3D model [5] (license: CC BY 4.0, by Artec 3D).

4.6. Model house for train free 3D print model

The complex shape model created in OBJ format shown in Fig. 27 was used for verification, and a portion of the cross-sectional view of Fig. 27 is shown in Fig. 28. The

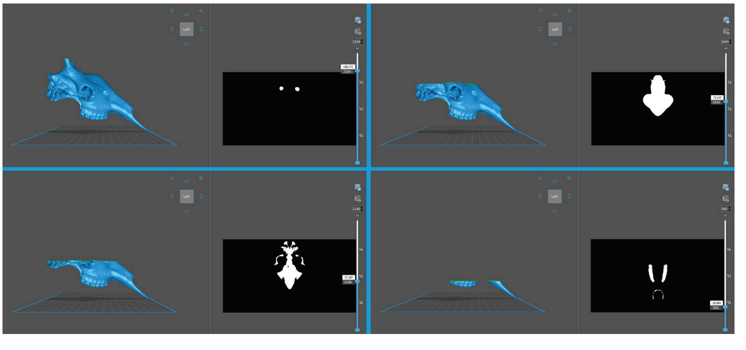


Fig. 26. Part of the cross-sectional view of the giraffe skull 3D model.

Tab. 6. Data size of various file formats and data size after compression by the proposed method (Giraffe skull: 744 647 vertices and 2 540 layers).

File type	File size [bytes]
OBJ (original) [5]	61 013 447
OBJ (after ZIP compression)	20 083 573
PWS	475 426 789
Photon	479 369 665
ZIP (2D image layers + metadata)	29 136 537
slc	45 616 214
phz	510 800 838
cws	27 715 959
pwx	44 495 436
svgx	81 846 149
Apply ZIP after the proposed method	7 718 206

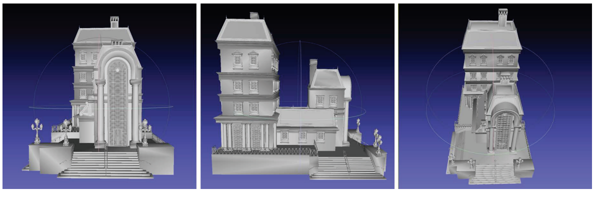


Fig. 27. Complex model house for train free 3D print model [39] (Royalty Free No Ai License from "cgtrader | General Terms and Conditions | 21A. Royalty Free License").

difference in data size for each file format is shown in Tab. 7. Note that this model contained 339 834 vertices and 3 371 layers.

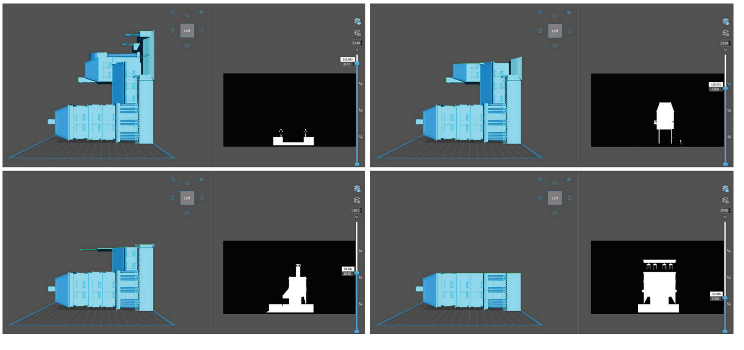


Fig. 28. Part of the cross-sectional view of the model house for train free 3D print model.

Tab. 7. Data size of various file formats and data size after compression by the proposed method (Model house for train free 3D print model: 339 834 vertices and 3 371 layers).

File type	File size [bytes]
OBJ (original) [39]	55 109 933
OBJ (after ZIP compression)	14 153 127
PWS	634 917 533
Photon	639 893 543
ZIP (2D image layers + metadata)	20 279 624
slc	24 930 638
phz	688 854 398
cws	19 942 696
pwx	68 048 184
svgx	47 438 712
Apply ZIP after the proposed method	1 184 268

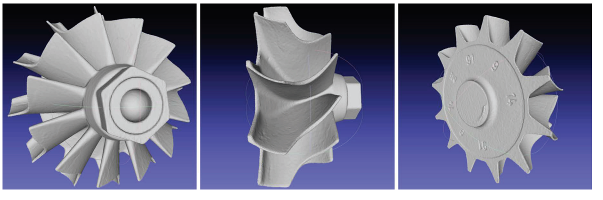


Fig. 29. Complex turbine 3D model [8] (license: CC BY 4.0, by Artec 3D).

4.7. Turbine

The complex shape model created in OBJ format shown in Fig. 29 was used for verification, and a portion of the cross-sectional view of Fig. 29 is shown in Fig. 30. The difference in data size for each file format is shown in Tab. 8. Note that this model contained 150 002 vertices and 1 856 layers.

4.8. Stool

The complex model created in OBJ format shown in Fig. 31 was used for verification, and a portion of the cross-sectional view of Fig. 31 is shown in Fig. 32. The difference in data size for each file format is given in Tab. 9. Note that this model contained 106 702 vertices and 4 237 layers.

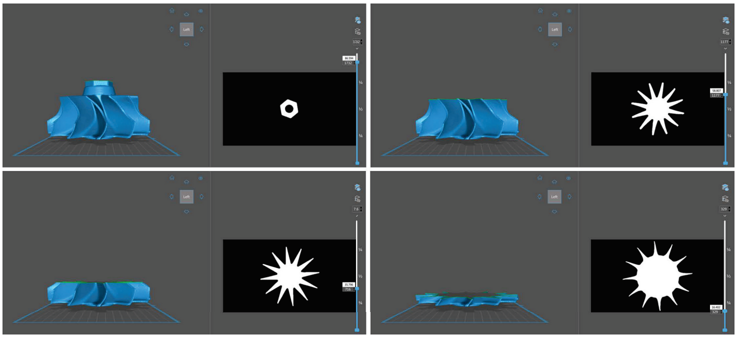


Fig. 30. Part of the cross-sectional view of the turbine 3D model.

Tab. 8. Data size of various file formats and data size after compression by the proposed method (Turbine: 150 002 vertices and 1 856 layers).

File type	File size [bytes]
OBJ (original) [8]	15 657 744
OBJ (after ZIP compression)	5 564 081
PWS	352 112 095
Photon	355 137 383
ZIP (2D image layers + metadata)	55 480 606
slc	27 272 182
phz	389 967 957
cws	53 156 143
pwx	50 088 630
svgx	50 166 142
Apply ZIP after the proposed method	22 291 844

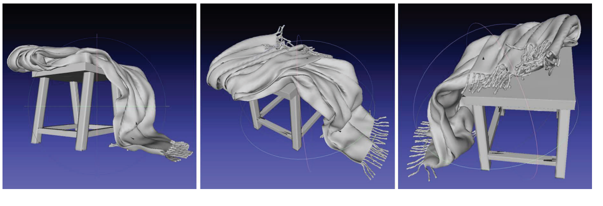


Fig. 31. Complex stool 3D model [22] (Royalty Free No Ai License from "cgtrader | General Terms and Conditions | 21A. Royalty Free License").

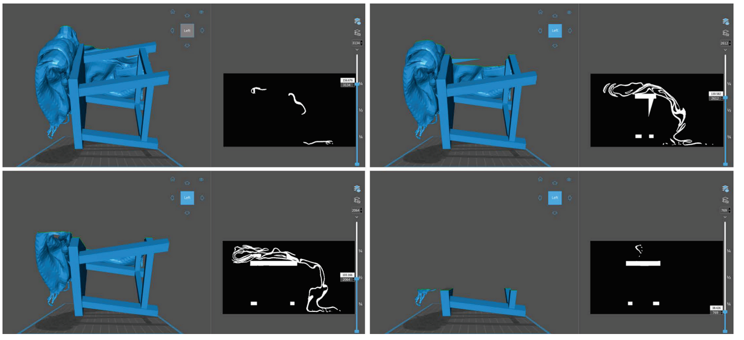


Fig. 32. Part of the cross-sectional of the stool 3D model.

Tab. 9. Data size of various file formats and data size after compression by the proposed method (Stool: 106 702 vertices and 4 237 layers).

File type	File size [bytes]
OBJ (original) [22]	15 616 745
OBJ (after ZIP compression)	4 096 359
PWS	810 316 247
Photon	816 776 523
ZIP (2D image layers + metadata)	109 642 159
slc	45 733 614
phz	901 592 107
cws	115 805 120
pwx	131 373 624
svgx	87 396 109
Apply ZIP after the proposed method	50 387 383

5. Discussion

For complex shapes with a large number of vertices, the proposed method obtained a higher compression ratio than the compared file formats. This is due to the large number of vertices required to print complex shapes, resulting in large original file sizes. However, we found that the proposed method was not necessarily effective for shapes with a small number of vertices. For files such as OBJ and STL, which create models based on vertex information, the file size is generally smaller when the number of vertices is small. Therefore, in the case of a file in a general printing format that is saved as a raster image for each layer, the file size will be larger than data with a small number of vertices. In other words, in such a file with a small number of vertices, the proposed method using the difference method for slice images inevitably results in a large file size.

On the other hand, as the 3D model becomes more complex, the number of vertices increases and the original file size also increases. In such cases, the proposed method using the difference method based on slice data would have been more effective. Thus, the proposed method is more effective for compressing large files that require a GPGPU when creating slice data.

Since this is an initial experiment, the slice data are output in ASCII format like OBJ files, but it is thought that the compression ratio can be further improved by using a binary format. In addition, this method does not consider the features of the image itself. It is considered that it is possible to further increase the compression ratio by utilizing shape similarity such as symmetry.

6. Conclusion

With the recent advancements in the case of LCD 3D printers, it has become possible to print extremely precise objects due to the high-resolution of the liquid crystal panels. As a result, the amount of data required for 3D printing has increased significantly. In contrast, 2D slice data, i.e., a cross-sectional view used for printing, tends to have continuity in the data (unlike general images). Thus, by focusing on such features, a high compression ratio can be obtained even with a method that uses a relatively simple difference.

In this study, we performed a preliminary verification experiment, where the compression results obtained by the proposed method were output as text data. In addition, to convert the data to a printable file format, the data obtained from the slicer were used without modification, and the final result summarized by ZIP was obtained as the compression ratio. We found that the proposed difference method is effective; thus, it is considered possible to improve the compression ratio further by applying a new compression method, e.g., a binary format.

Currently, the proposed method is effective for data with a large number of vertices that may require GPGPU processing for analysis. In other words, it is effective for large files; however, in future, we would like to improve the compression ratio for small files with a relatively small number of vertices.

References

[1] 3DWithUs. The Best Resin 3D Printers 2023 – Comparison & Innovations, 2023. `https://3dwithus.com/best-resin-3d-printers`, [Accessed: 17 Mar 2023].

[2] A. Ambrosi and M. Pumera. 3D-printing technologies for electrochemical applications. *Chemical Society Reviews*, 45(1):2740–2755, 2016. doi:10.1039/C5CS00714C.

[3] F. Arceo. The 8 best slicers for resin 3D printers (SLA, LCD, DLP). In: 3D Solved. `https://3dsolved.com/best-slicers-for-resin-3d-printers-sla-lcd-dlp/`, [Accessed: 11 Mar 2023].

[4] Area 515. Photonic3D. In: GitHub. https://github.com/area515/Photonic3D, [Accessed: 11 Mar 2023].

[5] Artec 3D. Giraffe skull. https://www.artec3d.com/3d-models/giraffe-skull, [Accessed: 11 Mar 2023].

[6] Artec 3D. Motorcycle engine HD. https://www.artec3d.com/3d-models/motorcycle-engine-hd, [Accessed: 11 Mar 2023].

[7] Artec 3D. Plaster cast of teeth. https://www.artec3d.com/3d-models/plaster-cast-teeth, [Accessed: 10 Mar 2023].

[8] Artec 3D. Turbine. https://www.artec3d.com/3d-models/turbine, [Accessed: 11 Mar 2023].

[9] AUTODESK Support. How to create a sketch of a cross-section of a 3D model in Fusion 360. AUTODESK, 2020. https://www.autodesk.com/support/technical/article/caas/sfdcarticles/sfdcarticles/How-to-create-a-sketch-of-a-cross-section-of-a-3D-Model-in-Fusion-360.html, [Accessed 15 Mar 2023].

[10] P. Bourke. SLC – SLiCe format. In: Paul Bourke personal/professional web pages. http://paulbourke.net/dataformats/slc/, [Accessed: 11 Mar 2023].

[11] E. Champion and H. Rahaman. Survey of 3D digital heritage repositories and platforms. *Virtual Archaeology Review*, 11(23):1–15, 2020. doi:10.4995/var.2020.13226.

[12] CHITUBOX. All-in-one SLA/DLP/LCD preprocessing slicer. In: CHITUBOX Docs. https://www.chitubox.com/en/page/chitubox-vs, [Accessed: 12 Mar 2024].

[13] A. O. Demirel. FDM vs SLA: 3D baskıda hangi teknolojiyi seçmelisiniz? In: 3dörtgen Blog, 2023. https://blog.3dortgen.com/fdm-vs-sla-3d-baskida-hangi-teknolojiyi-secmelisiniz/, [Accessed: 15 Jan 2024].

[14] S. Divakar and R. V. S. Satyanarayana. Comparison of matching algorithms for MST radar data. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(12):6271–6278, 2013. https://www.ijareeie.com/upload/2013/december/46_Comparison.pdf.

[15] P. Dudek. FDM 3D printing technology in manufacturing composite elements. *Archives of Metallurgy and Materials*, 58(4):1415–1418, 2013. doi:10.2478/amm-2013-0186.

[16] Ebrary.net. Additive manufacturing and 3D printing technology: Principles and applications, chapter: Format specifications. https://ebrary.net/158117/engineering/format_specifications, [Accessed: 11 Mar 2023].

[17] R. A. I. Elzain. Large-scale 3D printing – Market analysis. Technical Report 47, Constructor University Technical Reports, 9 Dec 2022. http://nbn-resolving.org/urn:nbn:de:gbv:579-opus-1011273.

[18] fabelar. Triceratops FREE printable Free 3D print model. In: cgtrader. https://www.cgtrader.com/free-3d-print-models/science/biology/triceratops-free-printable, [Accessed: 11 Mar 2023].

[19] FLASHFORGE. Explaining the types and features of stereolithography 3d printers! (in Japanese), 23 Jun 2022. https://flashforge.jp/benefit_list/about-stereolithography/, [Accessed: 11 Mar 2023].

[20] FormWare. Formware 3D: 3D printing slicing algorithms, their speed and applications, 2022. https://www.formware.co/article/SlicingSpeed, [Accessed: 11 Mar 2023].

[21] FormWare. Formware 3D: Powerful 3D Printing slice and support generation software, 2022. https://www.formware.co/slicer, [Accessed: 10 Mar 2023].

[22] giuliazocca. Stool with blanket 3D model – seat free 3D model. In: cgtrader. https://www.cgtrader.com/free-3d-models/furniture/chair/stool-with-blanket-3d-model-seat, [Accessed: 11 Mar 2023].

[23] X. Han, Z. Zhan, X. Song, and L. Cui. An additive manufacturing direct slicing algorithm based on a step model. *Electronics*, 11(10):1582, 2022. doi:10.3390/electronics11101582.

[24] S. Hill. How to slice STL files in Cura | Step by step guide. In: fabheads, 2022. https://www.crealitycloud.com/blog/tutorials/how-to-slice-stl-files-in-cura, Tutorial. [Accessed: 17 Mar 2023].

[25] Home3Ddo. Difference of stereolithography method "SLA, DLP, LCD, LFS, CLIP" 3D printer (in Japanese), 5 May 2023. https://home3ddo.blog.jp/3dprinter-sla, [Accessed: 14 Aug 2023].

[26] i-MAKER. What is the level of stacking for 3D printer optical fabrication? (in Japanese), 2021. https://i-maker.jp/blog/sla-stacking-pitch-16374.html, [Accessed: 11 Mar 2023].

[27] L. F. J. Hao and R. E. Williams. An efficient curvature-based partitioning of large-scale STL models. *Rapid Prototyping Journal*, 17(2):116–127, 2011. doi:10.1108/13552541111113862.

[28] D. Joldoshbek. King kong bust Free 3D print model. In: cgtrader. https://www.cgtrader.com/free-3d-print-models/games-toys/toys/king-kong-f0ab707c-9617-4d32-9f56-5f287f40bcd1, [Accessed: 10 Mar 2023].

[29] J. H. Kim, O. H. P. Baptista, A. P. Ayres, R. L. B. da Silva, J. F. Lima, et al. Accuracy comparison among 3D-printing technologies to produce dental models. *Applied Sciences*, 12(17):8425, 2022. doi:10.3390/app12178425.

[30] B. King, A. Rennie, and G. Bennett. An efficient triangle mesh slicing algorithm for all topologies in additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 112(3):1023–1033, 2018. doi:10.1007/s00170-020-06396-2.

[31] M. Kolla. What is the role of slicing in 3D printing? In: fabheads, 2021. https://fabheads.com/blogs/what-is-the-role-of-slicing-in-3d-printing/, [Accessed: 10 Mar 2023].

[32] A. A. Konta, M. Garcia-Piña, and D. R. Serrano. Personalised 3D printed medicines: Which techniques and polymers are more successful? *Bioengineering*, 4(4):79, 2017. doi:10.3390/bioengineering4040079.

[33] A. Kumar, A. Joshi, A. Kumar, and A. Mittal. Template matching application in geo-referencing of remote sensing temporal image. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7(2):201–210, 2014. doi:10.14257/IJSIP.2014.7.2.19.

[34] X. Lai and Z. Wei. Slicing algorithm and partition scanning strategy for 3D printing based on GPU parallel computing. *Materials*, 14(15):4297, 2021. doi:10.3390/ma14154297.

[35] A. O. Laplume, B. Petersen, and J. M. Pearce. Global value chains from a 3D printing perspective. *Journal of International Business Studies*, 47:595–609, 2016. doi:10.1057/jibs.2015.47.

[36] D. Lee. Template Matching. In: velog, as ldw200012.log, 12 Jul 2021. https://velog.io/@ldw200012/Template-Matching, [Accessed: 3 Apr 2023].

[37] Y. Lim. *Influence of Different Postprocessing Rinsing Agents on the Manufacturing Accuracy of Dental Models Printed by LCD Resin 3D Printer*. Master's thesis, Texas A & M University, 2022. [Accessed: 11 Mar 2023]. https://hdl.handle.net/1969.1/197435.

[38] M. Long. How to slice 3D printable files for filament and resin printers. In: electro{maker}, 2 Sep 2019. https://www.electromaker.io/tutorial/blog/how-to-slice-3d-printable-files-for-filament-and-resin-printers, Tutorial. [Accessed: 11 Mar 2023].

[39] T. Ly. Model house for train Free 3D print model. In: cgtrader. `https://www.cgtrader.com/free-3d-print-models/hobby-diy/other/model-house-for-train`, [Accessed: 11 Mar 2023].

[40] K. McHenry and P. Bajcsy. An overview of 3D data content, file formats and viewers. Tech. Rep. isda08-002, National Center for Supercomputing Applications, Image Spatial Data Analysis Group, 31 Oct 2008. `https://isda.ncsa.illinois.edu/peter/publications/techreports/2008/NCSA-ISDA-2008-002.pdf`.

[41] MiiCraft. User manual: MiiCraft 125. In: "MiiCraft | Support | User Manual | MiiCraft 125" `https://miicraft.com/user-manual/`. [Accessed: 11 Mar 2023].

[42] F. M. Mwema and E. T. Akinlabi. Basics of fused deposition modelling (FDM). In: *Fused Deposition Modeling: Strategies for Quality Enhancement*, pp. 1–15. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-030-48259-6_1.

[43] R. K. Müller. MSLA Anycubic Photon Mono 4K. XYZdims.com, 2023. `https://xyzdims.com/3d-printers/msla-anycubic-photon-mono-4k/`, [Accessed: 12 Aug 2023].

[44] T. Ngo, A. Kashani, G. Imbalzano, K. Nguyen, and D. Hui. Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. *Composites Part B: Engineering*, 143:172–196, 2018. doi:10.1016/j.compositesb.2018.02.012.

[45] H. Niitsuma and T. Maruyama. Sum of absolute difference implementations for image processing on FPGAs. In: *Proc. 2010 International Conference on Field Programmable Logic and Applications*, pp. 167–170. Milan, Italy, 31 Aug – 02 Sep 2010. doi:10.1109/FPL.2010.40.

[46] ocf81. Open Source 3D Printing – Deel 3, 2021. `https://gathering.tweakers.net/forum/list_messages/2059884`, [Accessed: 17 Mar 2023].

[47] S. Ourselin, R. Stefanescu, and P. Xavier. Robust registration of multi-modal images: Towards real-time clinical applications. In: T. Dohi and R. Kikinis, eds., *Medical Image Computing and Computer-Assisted Intervention – Proc. MICCAI 2002*, pp. 140–147. Springer Berlin Heidelberg, Tokyo, Japan, 25-28 Sep 2002. doi:10.1007/3-540-45787-9_18.

[48] A. V. Paramkusam and V. S. K. Reddy. An efficient fast full search block matching algorithm with SSD criterion. In: *Proc. 2011 Annual IEEE India Conference*, pp. 1–6. Hyderabad, India, 16-18 Dec 2011. doi:10.1109/INDCON.2011.6139485.

[49] G. N. Pham, S.-H. Lee, O.-H. Kwon, and K.-R. Kwon. A 3D printing model watermarking algorithm based on 3D slicing and feature points. *Electronics*, 7(2):23, 2018. doi:10.3390/electronics7020023.

[50] T. Pohlmann. Patent and litigation trends for 3D printing technologies. In: IAM, Law Business Research, 12 Mar 2019. `https://www.iam-media.com/article/patent-and-litigation-trends-3d-printing-technologies`, [Accessed: 11 Mar 2023].

[51] F. Schurig. Slicing Algorithms for 3D-Printing. In: *Proseminar: H4ck3r's D3l1ght*. Technische Universität München, TUM School of Computation, Information and Technology, Chair for Decentralized Information Systems and Data Management, 2015. `https://blog.hk-fs.de/wp-content/uploads/2015/08/Paper_Fabian_Schurig_3D_Printing.pdf`.

[52] N. Shahrubudin, T. Lee, and R. Ramlan. An overview on 3D printing technology: Technological, materials, and applications. *Procedia Manufacturing*, 35:1286–1296, 2019. doi:10.1016/J.PROMFG.2019.06.089.

[53] J. Slaymaker, J. Woolley, and S. Hirani. 3D printing in orthodontics: An introduction. *SVOA Dentistry*, 4(6):229–241, 2023. doi:10.58624/SVOADE.2023.04.0155.

[54] P. Stavropoulos, K. Tzimanis, T. Souflas, and H. Bikas. Knowledge-based manufacturability assessment for optimization of additive manufacturing processes based on automated feature recognition from CAD models. *The International Journal of Advanced Manufacturing Technology*, 122(2):993–1007, 2022. doi:10.1007/s00170-022-09948-w.

[55] L. D. Stefano and S. Mattoccia. Fast template matching using bounded partial correlation. *Machine Vision and Applications*, 13:213–221, 2003. doi:10.1007/s00138-002-0070-5.

[56] E. Taneva, B. Kusnoto, and C. A. Evans. 3D scanning, imaging, and printing in orthodontics. In: F. Bourzgui, ed., *Issues in Contemporary Orthodontics*. IntechOpen, Rijeka, 2015. doi:10.5772/60010.

[57] Tiger 3D. Tiger3D series: Operators manual, 2017. `https://www.romanoff.com/magenew/media/manuals/Tiger3D-Operators-Manual.pdf`, [Accessed: 11 Mar 2023].

[58] A. Wang, C. Zhou, Z. Jin, and W. Xu. Towards scalable and efficient GPU-enabled slicing acceleration in continuous 3D printing. In: *Proc. 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 623–628. Chiba, Japan, 16-19 Jan 2017. doi:10.1109/ASPDAC.2017.7858393.

[59] X. Zhang, G. Xiong, Z. Shen, Y. Zhao, C. Guo, et al. A GPU-based parallel slicer for 3D printing. In: *Proc. 2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 55–60. Xi'an, China, 20-23 Aug 2017. doi:10.1109/coase.2017.8256075.