Journal of Autonomous Intelligence (2024) Volume 7 Issue 1
doi: 10.32629/jai.v7i1.997

ORIGINAL RESEARCH ARTICLE

A Study of 3D model voxelization method for artificial intelligence learning

Byongkwon Lee

School of Media Contents, Department of Multimedia Major, Seowon University, Cheongju-si 28674, Korea; byongkwonlee53@gmail.com, sonic747@seowon.ac.kr

ABSTRACT

The 3D reconstruction technology which is one of various restoration technologies, implements and shapes 2D pixels of an object that actually exists in a 3D form. As an accurate 3D model for artificial intelligence learning information pre-processing is required. The pre-processing needs to accurately size information and coordinate information of a 3D object for artificial intelligence learning. Also, the 3D model data generated during the preprocessing can be represented as a point cloud, which is point-based coordinate data. In this study, the pixel data was analyzed by voxelizing the 3D model for artificial intelligence learning, and the 3D model data was digitized to form a mesh file. In this study, 3D modeling was done with a 3D modeling tool and the object was exported to STL. In addition, it was converted into mesh file data including 3D coordinate data in Python language on Google's colab platform. The mesh data created in this way is used in DataSet for artificial intelligence learning (CNN, RNN, GAN). Currently, there are many datasets for 2D artificial intelligence learning, but this study provided an opportunity to collect 3D artificial intelligence learning datasets. The research results can be used in the fields of robots, autonomous driving, games, and product design.

Keywords: 3D model voxelization; point cloud; 3D vision; AI 3D model learning; mesh 3D dataset

ARTICLE INFO

Received: 18 July 2023
Accepted: 28 July 2023
Available online: 30 October 2023

1. Introduction

A restoration of 3D buildings or objects is distinguished from a restoration of 2D images[1] or videos. The 2D reconstruction is based on two X, Y coordinates, but the 3D reconstruction[2,3] requires calculation of three-dimensional coordinates X, Y, Z. Currently, most search and tree gardening algorithms using artificial intelligence learn and utilize 2D data sets. However, 3D artificial intelligence learning has difficulties in artificial intelligence learning because there is only data specialized in a specific field and the 3D dataset required for learning is insufficient[4,5]. In this study, we studied how to convert a general 3D model (object) into a mesh file required for artificial intelligence learning. In the course of the research, the pagoda was modeled using a 3D model creation tool and exported as a numerical data STL file, which is artificial intelligence learning data. The exported 3D modeling data is used for 3D printing as an STL file. It contains coordinate data and numerical data. In this study, STL data was converted into mesh data and expressed in 3D space as points. The tool used for STL to mesh conversion used Python language for Google's colab-based platform[6,7]. The composition of this paper consists of research related to 3D modeling and voxelization in Section 2, research procedures and methods for voxelization in Section 3. In Section 4, there are the experiments and considerations on voxelization results, and in Section 5, there is the conclusion.

2. Literature review

2.1. Voxelization of 3D model

Voxel are extensions of the pixel (picture element), the smallest unit that makes up a 2D image into 3D. In other words, in image 1*1, it is expressed as 1*1*1 with depth information, and the smallest unit is called Voxel (Volume + Pixel). Units are not fixed and are user-definable. **Figure 1** shows the concept and application examples of voxels. In some documents, it is also expressed as a 3D Box or Cube.

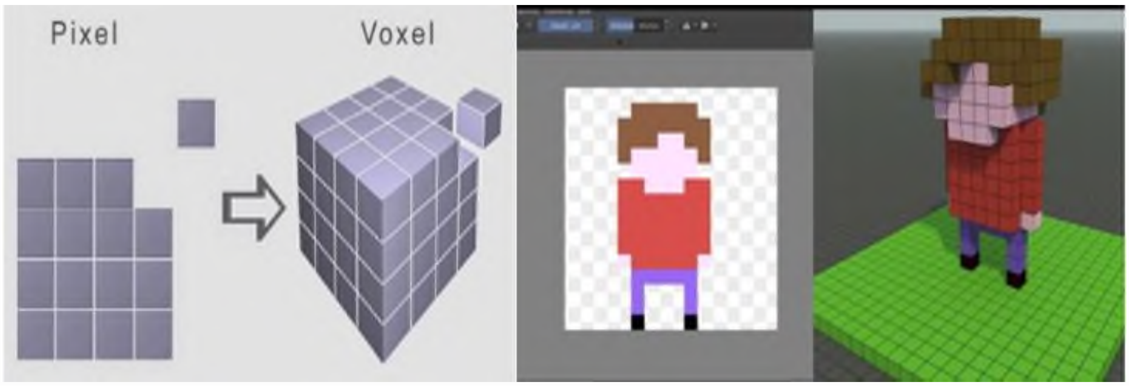


Figure 1. A concept (L) and application© for Voxelization 3D models.

Voxelization means converting a point cloud into voxels. PCL performs voxelization using a "Voxel Grid filter". The voxelization of the PCL method is shown in **Figure 2**. (a). Select Voxel size (=leaf_size) suitable for user definition. (b). Calculate the presence or absence of points (blue) within the leaf size from the center point of each voxel. (c). Calculate the center point (red) of the points and remove the remaining points[8].

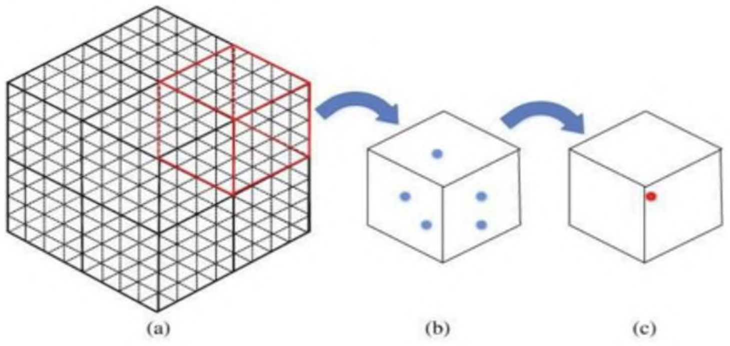


Figure 2. The process of the voxelization.

In the example of **Figure 2**, 5 points are expressed as one point (=Voxel). That is the size of the data has become 1/5. If the voxel unit (=leaf_size) is large, the amount of data can be further reduced. However, the expressiveness of the object will be reduced. After all, the most important thing in voxelization is to find the optimal unit (=leaf_size) in the trade-off relationship between computational load and object expressiveness.

2.2. Voxelization of GPU Nvidia

The basic concept of Nvidia GPU voxelization for converting scenes composed of triangular nets into regular voxel grid representations using GPU shaders[9]. The process for this is very simple. **Figures 3** and **4** describe this process visually[10].

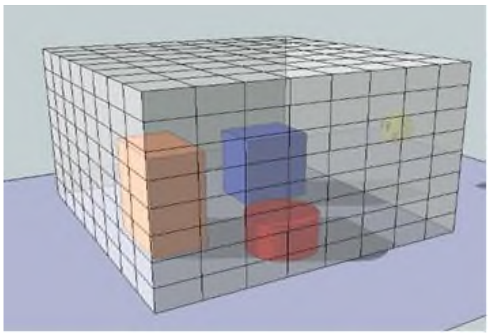


Figure 3. Orthogonal projection view.

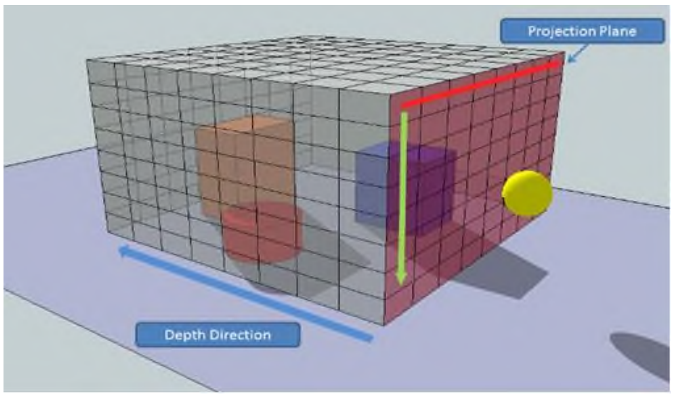


Figure 4. Render target and voxel grid.

3. An implementation of the voxelization for 3D model

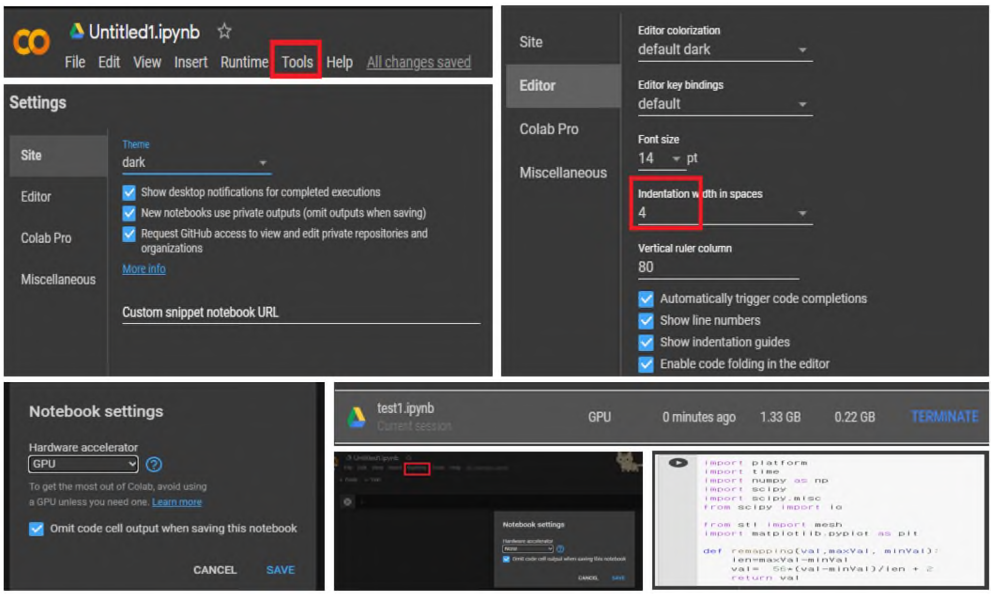
3.1. Configuration of environment

In this study, a voxelization experiment was conducted by finding a point of 3D Pagoda modeling data. **Table 1** is the development environment setting for the experiment. The used development language is Python. It can be seen that Tensorflow and Pytorch were used for artificial intelligence learning.

Table 1. Development environment.

DIV	Configuration
OS	Linux (Google Colab)
CPU	CPU: Intel (R) Xeon(R) CPU @ 2.30GHz (Dual-Core)
GPU	Nvidia Tesla T4.
Language	Python 3.9.12
CUDA version	CUDA 11.7
Platform	Pytorch 1.11.0 , Tensorflow-GPU 2.3.1

Google Colab is a free, cloud-based Jupyter notebook development environment. Internally, it is known to be composed of a technology stack of Colab + Google Drive + Docker + Linux + Google Cloud. Colab's features are free, easy to configure, multiple people can access it at the same time, and it's faster than a regular PC if you use the paid version[11,12]. However, the downside is that the maximum session duration is 12 hours. In other words, AI training must be completed within 12 hours. **Figure 5** is the environment setting for using Colab. Colab operates based on Google Drive, so you must be registered with Google. In addition, it has a great advantage because it can proceed with voxelization through the use of GPU. The coded voxelization program checks the transformation process while executing in units of modules.



Theme: dark

- [✓] Show desktop notifications for completed executions
- [✓] New notebooks use private outputs (omit outputs when saving)
- [✓] Request GitHub access to view and edit private repositories and organizations

More info

Custom snippet notebook URL

Editor colorization: default dark

Editor key bindings: default

Font size: 14 pt

Indentation width in spaces: 4

Vertical ruler column: 80

- [✓] Automatically trigger code completions
- [✓] Show line numbers
- [✓] Show indentation guides
- [✓] Enable code folding in the editor

Hardware accelerator: GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

- [✓] Omit code cell output when saving this notebook

test1.ipynb	GPU	0 minutes ago	1.33 GB	0.22 GB

```
import platform
import time
import numpy as np
import scipy
import scipy.misc
from scipy import io

from stl import mesh
import matplotlib.pyplot as plt

def remapping(val,maxVal, minVal):
    len=maxVal-minVal
    val= 56*(val-minVal)/len + 2
    return val
```

Figure 5. Colab environment setup for voxelization.

3.2. Convert to MAT from STL mesh data

In MATLAB version 5, a MAT consists of a 128-byte header and one or more data elements. Each data element consists of an 8-byte tag and data from the element. Tags specify the number of bytes in the data element and how they are interpreted. This means that bytes must be read as 16-bit values, 32-bit values, floating-point values, or other data types. Version 5 MAT file formats allow fast access to individual data elements within a MAT file using tags[13,14]. **Figure 6** shows MAT format.

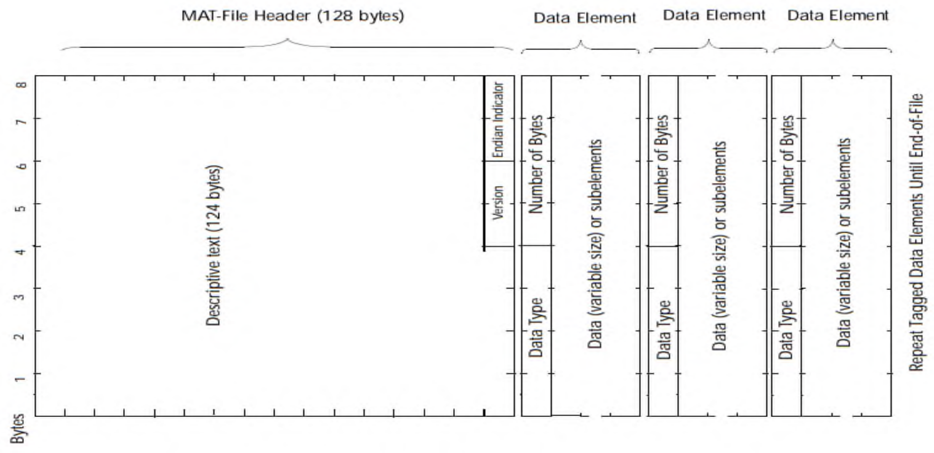

MAT-File Header (128 bytes)
Data Element
Data Element
Data Element
Bytes
1
2
3
4
5
6
7
8
Descriptive text (124 bytes)
Version
Endian Indicator
Data Type
Number of Bytes
Data (variable size) or subelements
Repeat Tagged Data Elements Until End-of-File

Figure 6. MATLAB version MAT-File Format.

The voxelization process for AI training is divided into four steps. **Figure 7** is a flowchart of the process. In **Figure 7**, (1) is the process of modeling a 3D object and creating a mesh using the graphic authoring tool Blender. (2) is the process of exporting (*.stl) the created 3D model to include numerical data. (3) The STL file is converted to a Matlab file (*.mat), which is used by a Python program to convert the numerical data. This step is divided into four sub-steps. (3.1) loads the program through Google's colab and loads the STL file created in (3.2). (3.3) Loading the imported mesh data into memory to rearrange the X, Y, Z coordinates. In (3.4), the loaded STL data is voxelized into triangles. (3.4.1) Check if the cube diagonals cross the triangle in the cube. (3.4.2) Check if point is inside triangle. (3.4.3) Match the triangle data. Finally, generate and save MAT data in (4).

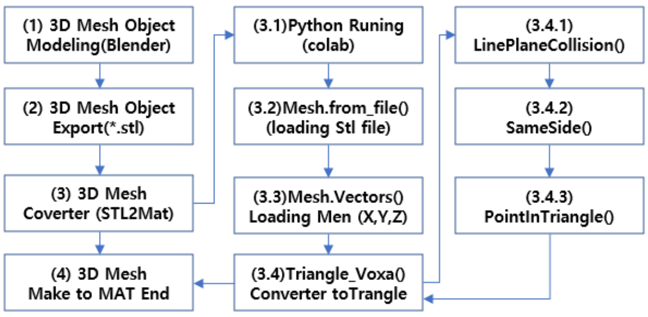
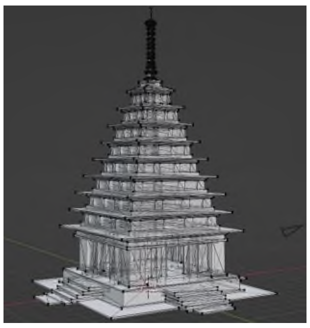
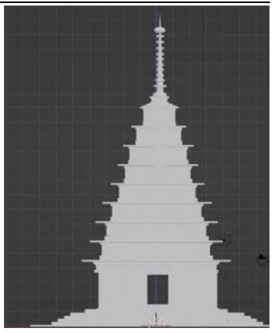
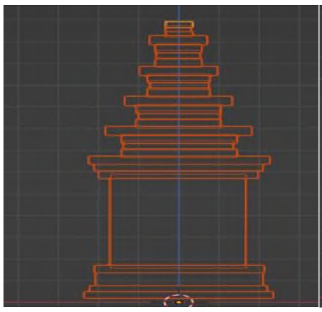
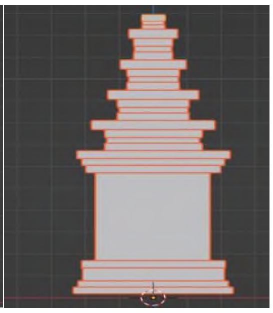
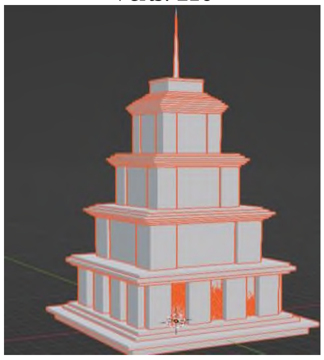
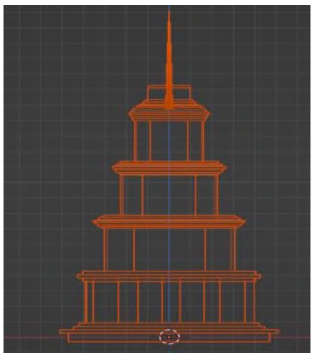
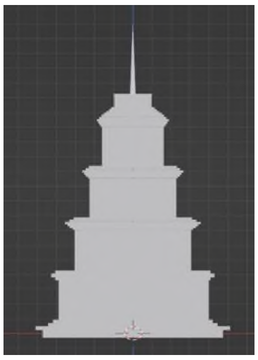

(1) 3D Mesh Object Modeling(Blender)
(2) 3D Mesh Object Export(*.stl)
(3) 3D Mesh Coverter (STL2Mat)
(4) 3D Mesh Make to MAT End
(3.1)Python Runing (colab)
(3.2)Mesh.from_file() (loading Stl file)
(3.3)Mesh.Vectors() Loading Men (X,Y,Z)
(3.4)Triangle_Voxa() Converter toTrangle
(3.4.1) LinePlaneCollision()
(3.4.2) SameSide()
(3.4.3) PointInTriangle()

Figure 7. Converter Flowchart to MAT from STL.

3.3. 3D mesh object modeling

3D Mesh Object Modeling is to model an object using the 3D graphics tool blender tool[15]. Modeling targets were Korean traditional pagodas from Goryeo, Silla, and Baekjea. A total of three pagodas were modeled, and **Table 2** shows the number of points and faces of the 3D modeled Pagoda. It also shows Perspective, Front and Shading views.

Table 2. Modeled points and faces of 3D pagoda.

Index	Perspective	Front	Shading
1	 Verts: 8,301	Face: 15,818	 Tris: 15,818
2	Verts: 216	 Face: 162	 Tris: 324
3	 Verts: 473	 Face: 363	 Tris: 722

The recreated 3D modeling object is saved as an OBJ file through the Export function. The exported OBJ file proceeds with the voxelization process.

3.4. Voxelization of 3D models

The purpose of this section is to find a data set optimized for artificial intelligence learning by voxelizing the 3D object modeled in section 3.3 with a Python program.

Table 3 is a representative stone pagoda in Korea, which is a voxel of the Mireuksa Temple Site in Gyeongju. Modeling is the number of vertices 8,301. In the voxelization process, the size of the X, Y, and Z axes (l*m*n) was changed to find the optimal voxelization similar to the original data. In this study, the values 1, 4, 8, 16, 34, 64, 128, and 256 were changed based on the height coordinate Y axis. It was confirmed that when the Y-axis value is confined to 128 voxels, it is voxelized similarly to the original data. However, in the case of 180*256*181/Voxels: 713,000, it was similar to the original data, but the data capacity was over 300 Mb, so it was confirmed that it was not suitable for artificial intelligence learning. In the case of initial modeling, it is judged that the model should proceed while minimizing the number of vertices.

Table 3. Index Number 1 3D Model in Section 3.3.

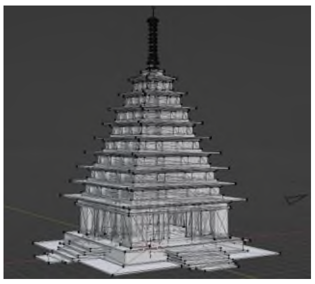
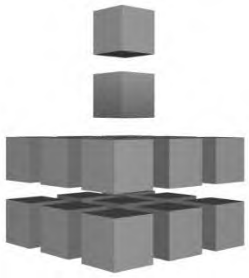
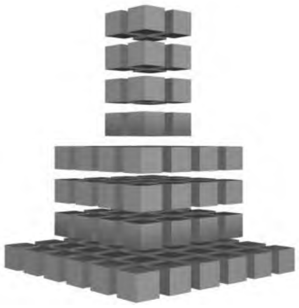
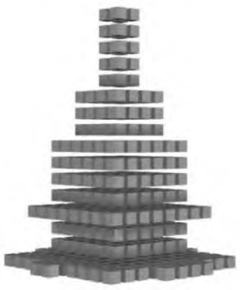
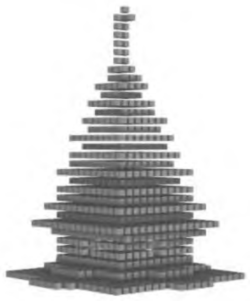
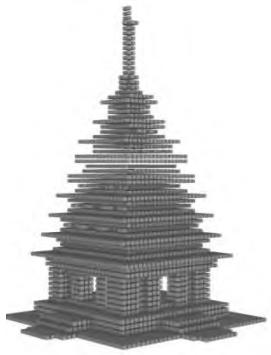
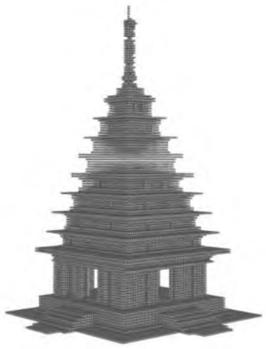
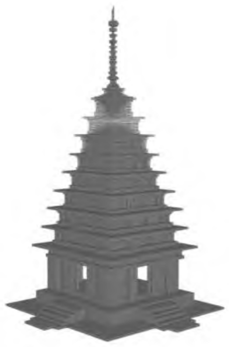
 original model Verts: 8301	1*1*1 Voxels: 1	 3*4*3 Voxels: 20
 6*8*6 Voxels: 100	 12*16*12 Voxels: 460	 23*34*23 Voxels: 2450
 45*64*46 Voxels: 15,277	 90*128*91 Voxels: 102,763	 180*256*181 Voxels: 713,000

Table 4. Index Number 2 3D Model in Section 3.3.

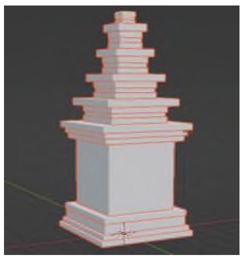
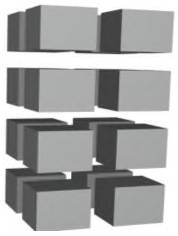
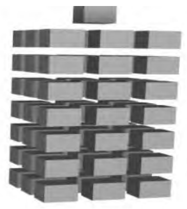
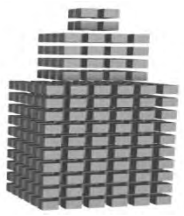
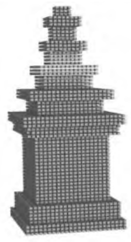
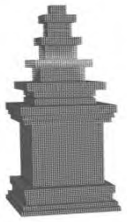
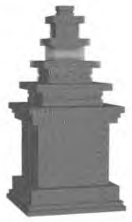
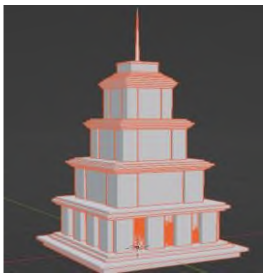
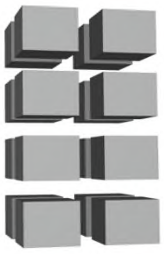
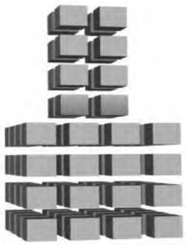
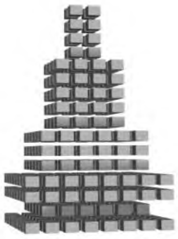
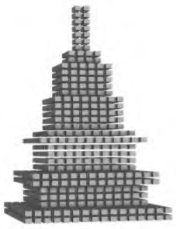
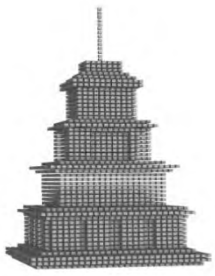
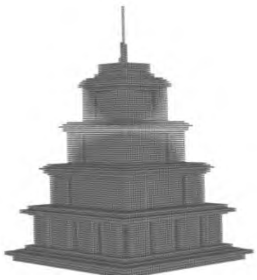
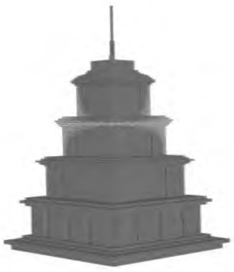
 original model Verts: 216	1*1*1 Voxels: 1	 2*4*2 Voxels: 16
 3*8*3 Voxels: 64	 6*16*6 Voxels: 432	13*34*13 Voxels: 3116
 24*64*24 Voxels: 3116	 48*128*48 Voxels: 130,222	 95*256*95 Voxels: 820,280

Table 4 is the voxelization of the second model of Index number in Section 3.3. The number of vertices is 216. This model voxelization was reconstructed 1, 4, 8, 16, 34, 64, 128, 256 based on the Y axis. The first similar to the original 3D model is 24*64*24. Voxels: 3116. It was confirmed that it is optimized when modeling small vertices with 64 generated voxels.

Table 5 is the voxelization of the 3rd model of Index number in Section 3.3. The number of vertices is 473. This model voxelization was reconstructed 1, 4, 8, 16, 34, 64, 128, 256 based on the Y axis. The first similar to the original 3D model is 61*128*61 Voxels:128,078. It was confirmed that the number of generated voxels was optimized to 128,078. As a result of the three experiments presented above, when collecting a dataset for AI learning through voxelization, it is good to minimize the number of vertices of the model. However, it was confirmed that 256 cells in the Y-axis are good when 3D data is converted into 3D voxels due to a lack of similarity with the original data.

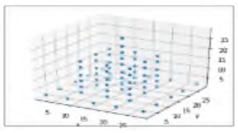
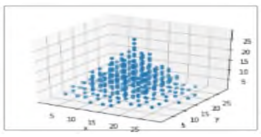
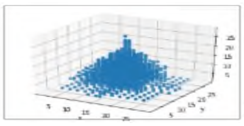
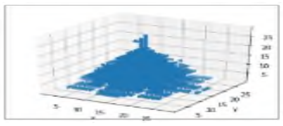
Table 5. Index Number 3 3D Model in Section 3.3.

 original model Verts: 473	1*1*1 Voxels: 1	 2*4*2 Voxels: 16
 4*8*4 Voxels: 80	 8*16*8 Voxels: 432	 16*32*16 Voxels: 2540
 31*64*31 Voxels: 17,691	 61*128*61 Voxels: 128,078	 122*256*122 Voxels: 128,078

4. Result and discussion

Blender was used as a 3D modeling tool to create a data set for artificial intelligence learning. In this study, when producing a 3D model, it should be made to express the characteristics of the object most effectively while minimizing the number of vertices and faces. In addition, the expression of numerical data for voxelization must be preceded by an MAT data conversion program in the case of an STL file used. In this study, Google's Colab was used to easily set the environment and system environment. In order to find the optimal cell size (height) for the voxel converted to MAT data, the experiment was conducted while changing the Y-axis value to 1, 4, 8, 16, 34, 64, 128, 256. As a result of this experiment, it was confirmed a voxel that is most similar to the original data (shape) and contains 128 heights with a small number of vertices. **Table 6** records the conversion time to point cloud form for model in Section 3.3.

Table 6. Voxelization transformation time.

Model	Vector	time	Result
	1	60 s	
	2	80 s	
	3	112 s	
	5	141 s	

5. Conclusion and future work

AI training on data with 3D coordinates requires a large number of datasets. Most of the existing data is difficult to utilize as a 2D data set. In this study, a 3D object was created using a 3D modeling tool and a dataset for artificial intelligence learning was obtained through voxelization. However, voxelization has a problem in that it must be transformed into a form most similar to the original data. Therefore, in this study, the optimal voxelization was found while adjusting the size of the voxel. As a result of the study, optimal voxelization should first be produced with the minimum number of vertices in the initial modeling work. In addition, it was confirmed that the shape of the original data can be maintained if the cell size of the Y-axis corresponding to the height is set to 128 during voxelization.

Future research projects require research on voxelized data sets such as point cloud type GAN. As a field of application of this study, it is considered that it will be used for restoration of cultural assets and restoration of 3D objects.

Conflict of interest

The author declares no conflict of interest.

References

1. Brown MS, Sun M, Yang R, et al. Restoring 2D content from distorted documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2007; 29(11): 1904–1916. doi: 10.1109/TPAMI.2007.1118
2. Banerjee A, Zacur E, Choudhury RP, Grau V. Automated 3D whole-heart mesh reconstruction from 2D cine MR slices using statistical shape model. *Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)* 2022; 2022: 1702–1706. doi: 10.1109/EMBC48229.2022.9871327
3. Lv S, Zhu Y, Ni H, et al. Teapot three-dimensional geometrical model reconstruction based on reverse engineering and rapid prototyping technology. In: Proceedings of the 2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE); 2018; Huhhot, China. pp. 180–184.
4. Sabbella DS, Singh A, Maheswari G. Artificial intelligence in 3D CAD modelling. In: Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE); 2020; Vellore, India. pp. 1–5.
5. Singh A, Srivastava AP, Bhardwaj G, et al. Methods to detect an event using artificial intelligence and machine learning. In: Proceedings of the 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM); 2022; London, United Kingdom. pp. 297–301.

6. Ali I, Khan A, Waleed M. A google colab based online platform for rapid estimation of real blur in single-image blind deblurring. In: Proceedings of the 2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI); 2020; Bucharest, Romania. pp. 1–6.
7. Firigato JON, Junior JM, Gonçalves WN, Bacani VM. Deep learning and google earth engine applied to mapping eucalyptus. In: Proceedings of the 2021 IEEE International Geoscience and Remote Sensing (IGARSS); 2021; Brussels, Belgium. pp. 4696–4699.
8. Koide K, Yokozuka M, Oishi S, Banno A. Voxelized GICP for fast and accurate 3D point cloud registration. In: Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA); 2021; Xi'an, China. pp. 11054–11059.
9. Potluri S, Goswami A, Rossetti D, et al. GPU-Centric communication on NVIDIA GPU clusters with InfiniBand: A case study with OpenSHMEM. In: Proceedings of the 2017 IEEE 24th International Conference on High Performance Computing (HiPC); 18–21 December 2017; Jaipur, India. pp. 253–262.
10. Choquette J, Gandhi W. NVIDIA A100 GPU: Performance & innovation for GPU computing. In: Proceedings of the 2020 IEEE Hot Chips 32 Symposium (HCS); 16–18 August 2020; Palo Alto, California. pp. 1–43.
11. Canesche M, Bragança L, Neto OPV, et al. Google colab CAD4U: Hands-on cloud laboratories for digital design. In: Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS); 22–28 May 2021; Daegu, Korea. pp. 1–5.
12. Shariar S, Hasan KMA. GPU accelerated indexing for high order tensors in google colab. In: Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP); 5–7 June 2020; Dhaka, Bangladesh. pp. 686–689.
13. Zhang Z, Lan W, Xin J, Li Q. A hybrid compress method of STL Mesh for realtime VR visualization. In: Proceedings of the 2020 7th International Conference on Information Science and Control Engineering (ICISCE); 18–20 December 2020; Changsha, China. pp. 27–30.
14. Xie T, Liu Z, Li J, et al. Development of launch vehicle shape design software based on parameterization method. In: Proceedings of the 2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE); 5–7 August 2022; Guangzhou, China. pp. 100–104.
15. Patoli MZ, Gkion M, Al-Barakati A, et al. An open source Grid based render farm for Blender 3D. In: Proceedings of the 2009 IEEE/PES Power Systems Conference and Exposition; 15–18 March 2009; Seattle, WA, USA. pp. 1–6.