Mixar
The AI-Native software stack for 3D

REPORT

ON

Mesh Normalization, Quantization, and Error Analysis

BY

SAKET BISHNU

RA2211027010059

Sb0558@srmist.edu.in

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

B.Tech – Computer Science and Engineering (Big Data Analytics Specialization)

1. INTRODUCTION

In recent years, 3D graphics and artificial intelligence have changed the way machines understand spatial data. Applications like autonomous navigation and AR/VR, along with intelligent design tools such as SeamGPT, now require AI systems to process and interpret complex 3D meshes accurately and consistently.

A 3D mesh shows an object's geometry through linked vertices and faces. However, raw meshes often come from various sources, such as 3D scanners, CAD models, or photogrammetry. Each source has its own coordinate scale, orientation, and resolution. These inconsistencies make it hard to use raw meshes directly in AI pipelines, leading to unreliable learning performance and biased predictions.

To address this, data preprocessing is an essential first step. Preprocessing means normalizing vertex coordinates to a fixed range, quantizing them into discrete bins for more compact storage, and then reconstructing the original mesh to see how much geometric information is lost.

This project focuses on creating a complete preprocessing workflow for 3D meshes, which includes:

- Loading .obj files and analyzing vertex data.
- Applying normalization using Min-Max and Unit Sphere methods.
- Quantizing coordinates with fixed-precision bins (1024 levels).
- Reversing the transformations through dequantization and denormalization.
- Measuring and visualizing reconstruction errors, such as MSE and MAE.

By following these steps, the project aims to ensure that 3D meshes of different scales can be presented in a consistent, standardized numerical format. This makes them suitable for later AI and machine learning tasks.

Ultimately, this assignment shows how important clean and measurable 3D data preprocessing is as a foundation for training smart geometric AI models like SeamGPT.

2. Scope of Work

The main goal of this assignment was to create a strong preprocessing pipeline for .obj mesh files.

The scope includes:

- Loading and checking 3D mesh vertex data.
- Using normalization techniques to adjust all coordinates to a common scale.
- Quantizing the normalized data with a fixed bin size of 1024.
- Reversing the process with dequantization and denormalization.
- Calculating error metrics, including MSE and MAE.
- Visualizing and comparing results for different normalization methods.

3. Key Concepts

3.1 Mesh Structure

A 3D mesh includes:

Vertices: 3D coordinates (x, y, z)

Faces: Triangles formed by connecting vertices. Only vertices were processed in this experiment because they define the geometric structure.

3.2 Normalization

Normalization makes sure that all meshes, no matter their original size, fit within a consistent range.

The methods used were:

Min-Max Normalization

$x' = x - xmin / xmax - xmin$

This scales each axis independently into the [0, 1] range.

Unit Sphere Normalization $x' = x - \mu / r$

This centers the mesh around its mean (centroid) and scales it to fit inside a sphere with a radius of 1.

3.3 Quantization

Quantization compresses floating-point coordinates into discrete bins. This reduces precision while keeping the structure intact. For this task, 1024 bins were used:

$q = int(x' \times (nbins - 1))$

To reconstruct:

$x'' = (nbins - 1)q$

This step is important for efficient storage and transmission of 3D geometry.

3.4 Error Metrics

To assess how much information is lost during normalization and quantization, two metrics were calculated:

Mean Squared Error (MSE): This measures the squared average difference between the original and reconstructed vertices.

Mean Absolute Error (MAE): This measures the average absolute difference.

Both metrics were computed per-axis and averaged over all vertices.

4. Implementation Details

4.1 Tools and Libraries Used

- Python 3.10 (Anaconda Environment: meshproc)
- NumPy – for numerical operations
- Matplotlib – for visualization
- Trimesh / Open3D – for loading and inspecting meshes

4.2 Folder Structure

mesh_preprocessing/

data/

- 8samples/

outputs/

- [meshname]_stats.json
- [meshname]_normalized_minmax.obj
- [meshname]_normalized_unitsphere.obj
- [meshname]_reconstructed_minmax.obj
- [meshname]_reconstructed_unitsphere.obj
- [meshname]_error_minmax.png
- [meshname]_error_unitsphere.png
- summary.json

mesh_preprocessing.py

summ.py

README.md

5. Results and Discussion

Mesh	**Min–Max MSE**	**Unit Sphere MSE**	**Min–Max MAE**	**Unit Sphere MAE**
Branch	2.35×10^{-6}	6.55×10^{-6}	0.00220	0.00382
Cylinder	2.39×10^{-6}	7.72×10^{-6}	0.00183	0.00415
Explosive	3.72×10^{-7}	1.10×10^{-6}	0.00083	0.00157
Fence	4.69×10^{-7}	1.44×10^{-6}	0.00082	0.00182
Girl	6.18×10^{-7}	1.13×10^{-6}	0.00111	0.00160
Person	2.29×10^{-6}	5.08×10^{-6}	0.00203	0.00335
Table	4.50×10^{-7}	1.49×10^{-6}	0.00093	0.00183
Talwar	3.88×10^{-7}	1.89×10^{-6}	0.00068	0.00209

Average Errors (across 8 meshes):

- Min–Max Normalization:
  - MSE = 1.165×10^{-6}
  - MAE = 0.00130
- Unit Sphere Normalization:
  - MSE = 3.302×10^{-6}
  - MAE = 0.00253

6. CONCLUSION

The completion of this assignment represents the successful development and execution of a complete mesh preprocessing pipeline. This pipeline standardizes and prepares 3D mesh data for AI applications. The implementation included all essential stages of the preprocessing workflow, such as mesh loading and inspection, normalization, quantization and dequantization, and error computation with visualization. First, we analyzed the .obj meshes to extract and understand their vertex data. Then, we applied two distinct normalization techniques: Min-Max normalization and Unit Sphere normalization. These methods ensured that meshes with different scales and orientations were brought into a consistent coordinate range suitable for further processing.

Next, we implemented quantization to convert continuous vertex coordinates into 1024 bins. This approach effectively compressed the data while maintaining its geometric integrity. The dequantization and denormalization steps then reconstructed the original meshes, allowing for direct comparison between the original and transformed data. We carried out quantitative evaluation using Mean Squared Error (MSE) and Mean Absolute Error (MAE). The results showed that Min-Max normalization combined with 1024-bin quantization produced the most accurate reconstruction, achieving an average MSE of 1.165×10^{-6} and an MAE of 0.0013. This indicates that Min-Max normalization effectively preserves vertex-level precision across different mesh geometries.

While Unit Sphere normalization resulted in slightly higher errors, it remained useful for cases requiring rotation and orientation invariance. This property is important in 3D modeling and AI-driven geometry understanding. Overall, the developed preprocessing pipeline ensures that raw, unstandardized 3D data can transform into a consistent, quantized, and measurable format. Such a pipeline improves data quality and provides a solid foundation for systems like SeamGPT, which rely on structured 3D mesh data for accurate learning, reasoning, and generation of geometric insights.