Assignment Title: Mesh Normalization, Quantization, and Error Analysis
Total Marks: 100

Context

Our company works in the 3D graphics and AI space — building intelligent systems like SeamGPT, which understands 3D meshes and generates UV mappings, materials, and more.

Before any AI model can learn from 3D meshes, the data must be clean, normalized, and quantized properly. This ensures that different meshes have consistent coordinate ranges and formats.

This assignment focuses on understanding and implementing **data preprocessing for 3D meshes**, which is a fundamental step in our research workflow.

You will not train or fine-tune any AI models. Instead, you will write code that prepares 3D data in a clean, consistent, and measurable way.

Scope of Work

You are given a folder containing `.obj` mesh files. (here)
Your task is to:

1. Load and understand the mesh data.
2. Apply normalization (bringing coordinates into a standard range).
3. Apply quantization (discretizing coordinates into bins).
4. Reverse the transformations (dequantize, denormalize).
5. Measure and visualize the differences between the original and transformed meshes.

Key Concepts

1. Mesh
A mesh is a collection of:

- Vertices — points in 3D space (x, y, z)
- Faces — surfaces connecting these vertices (usually triangles)

In this assignment, we'll only focus on vertices.

---

2. Normalization
Normalization ensures that different meshes (big or small) fit into a standard range.

Common normalization methods include:

- **Min–Max Normalization:**
  Bring all vertex coordinates into a range like [0, 1] or [-1, 1].
  Formula: $x' = (x - x_{min}) / (x_{max} - x_{min})$
- **Z-Score Normalization:**
  Center the mesh and scale it by standard deviation.
  Formula: $x' = (x - \mu) / \sigma$
- **Unit Sphere Normalization:**
  Scale the mesh so that all vertices fit inside a sphere of radius 1.

---

3. Quantization
Quantization reduces continuous values into discrete bins.

If we choose a bin size of 1024:

- Each axis (x, y, z) can take integer values between 0 and 1023.
- It's like "compressing" the mesh representation while keeping its structure.

Formulas:

Quantization (for normalized coordinates in [0, 1]):
$q = \text{int}(x' \times (n_{bins} - 1))$

Dequantization (to recover):
$x'' = q / (n_{bins} - 1)$

---

4. Error Measurement
To check how good your transformations are, compute **Mean Squared Error (MSE)** or **Mean Absolute Error (MAE)** between:

- Original mesh vertices
- Reconstructed (dequantized + denormalized) vertices

This will help you evaluate information loss after quantization and normalization.

---

5. Visualization (Optional)
You can visualize meshes in Blender or use a Python library such as `open3d`.

Example idea (not mandatory to use Blender if unavailable):

- Before normalization
- After normalization
- After quantization and reconstruction

Visuals help verify if the structure of the mesh is preserved.

Assignment Tasks (Total: 100 Marks)

Task 1: Load and Inspect the Mesh (20 Marks)

Goal: Understand and visualize 3D mesh data.

Steps:

1. Load `.obj` meshes using libraries such as `trimesh` or `open3d`.
2. Extract vertex coordinates (x, y, z) as a NumPy array.
3. Print basic statistics:
    - Number of vertices
    - Minimum, maximum, mean, and standard deviation per axis
4. Visualize the original mesh (optional).

Expected Output:
Printed statistics and optionally a screenshot or rendered view of the mesh.

Task 2: Normalize and Quantize the Mesh (40 Marks)

Goal: Convert the mesh into a standard numerical form.

Steps:

1. Implement two normalization methods of your choice (for example, Min–Max and Unit Sphere).
2. For each method:
    - Normalize all vertex coordinates.
    - Quantize with a bin size of 1024.
3. Save the quantized mesh as `.ply` or `.obj`.
4. Visualize both the normalized and quantized meshes.

Deliverables:

- Two normalized meshes
- Two quantized meshes
- Short written comparison: which normalization method preserves the mesh structure better?

Task 3: Dequantize, Denormalize, and Measure Error (40 Marks)

Goal: Check how much information is lost after processing.

Steps:

1. For each normalization and quantization combination:
    - Dequantize to recover normalized coordinates.
    - Denormalize to recover the original scale.
2. Compute Mean Squared Error (MSE) or Mean Absolute Error (MAE) between:
    - Original mesh vertices
    - Reconstructed vertices
3. Visualize reconstructed meshes.
4. Plot reconstruction error per axis (x, y, z) using Matplotlib.
5. Write a short conclusion (5–10 lines):
    - Which normalization and quantization combination gives the least error?
    - What pattern do you observe?

Deliverables:

- Plots of error metrics
- Screenshots of reconstructed meshes
- Written analysis

Marking Scheme

Task	Description	Marks
Task 1	Mesh loading and inspection	20
Task 2	Normalization and quantization	40
Task 3	Reconstruction and error analysis	40
Total		**100**

Submission Guidelines

- Submit a single ZIP file containing:
    - Your Python scripts or notebook
    - Output meshes
    - Visualizations and plots
    - A short README (explaining how to run code) with your observations
    - Final Pdf report to summarize the end-to-end observations and results
- You may use the following libraries:
  `Python`, `NumPy`, `Open3D`, `Trimesh`, `Matplotlib`, or `Blender`.
- GPUs are not required — all steps can be executed on CPU.

Bonus Task (Optional – 30 Marks)

Advanced Mesh Understanding and Research Challenge

Choose **any one** of the following sub-tasks. Each focuses on a deeper concept related to how SeamGPT and similar systems process 3D data.

Option 1: Seam Tokenization Prototype

Goal: Prototype how seams of a 3D mesh could be represented as discrete tokens — a step toward SeamGPT-style processing.

Steps:

1. Identify mesh seams (edges where UV mappings break).
2. Propose a token encoding scheme that can represent seam structure sequentially.
3. Show a simple example of encoding and decoding seams using your format.

Deliverables:

- Code or pseudocode for encoding/decoding
- Example token sequence
- Short explanation (5–10 lines) connecting this idea to mesh understanding

---

Option 2: Rotation and Translation Invariance + Adaptive Quantization

Goal:

Implement a normalization and quantization pipeline that is robust to mesh transformations and adapts to local geometric density.

Steps:

1. Generate multiple randomly rotated or translated versions of the same mesh.
2. Normalize each version using your method so that results remain consistent across transformations.
3. Analyze the vertex distribution and compute local density or variance to determine adaptive quantization bin sizes.
4. Quantize the normalized meshes using variable bin sizes — smaller bins in dense areas, larger bins in sparse ones.
5. Dequantize and denormalize the meshes to reconstruct the originals.
6. Measure and plot reconstruction error across transformed versions and compare against uniform quantization.

Deliverables:

- Normalized and quantized meshes for different orientations
- Error plots (uniform vs. adaptive quantization)
- Comparison table showing reconstruction error across methods
- Short written analysis discussing:
    - Invariance of normalization under transformations
    - Effectiveness of adaptive quantization in reducing information loss

---

Marking (30 Marks):

- Concept originality – 10 marks
- Implementation depth – 10 marks
- Clarity of explanation and results – 10 marks

Sample Code Snippet (For Reference Only)

This code is only to help you get started:

```
import trimesh
import numpy as np

# Load mesh
mesh = trimesh.load('path_to_mesh.obj')
vertices = mesh.vertices

# Min–Max normalize
v_min, v_max = vertices.min(axis=0), vertices.max(axis=0)
normalized = (vertices - v_min) / (v_max - v_min)

# Quantize
bins = 1024
quantized = np.floor(normalized * (bins - 1)).astype(int)

# Dequantize
dequantized = quantized / (bins - 1)

# Denormalize
reconstructed = dequantized * (v_max - v_min) + v_min

# Compute error
```

```
mse = np.mean((vertices - reconstructed) ** 2)
print(f"MSE: {mse:.6f}")
```

End Note

This assignment is designed to help you:

- Understand how 3D data is prepared before training AI models.
- Gain practical experience with normalization, quantization, and error analysis.
- Observe how preprocessing choices affect data quality and reconstruction accuracy.

By completing this, you'll gain an appreciation of how data pipelines like SeamGPT prepare 3D meshes for intelligent learning and generation.
