Update 1: Data Loading and Preprocessing

Govind Joshi

April 15, 2025

1 Data Acquisition

We obtained the dataset by contacting the authors of RigNet (Zhan Xu *et al.*). They provided two folders. One containing:

- **Unprocessed FBX files**: raw character meshes.
- **Processed meshes**: FBX files unpacked into obj files containing mesh data and txt files containing rig info. There are other files but they are irrelevant to our task. Each mesh has 1–5k vertices, and its longest dimension is normalized to 1.
- **Split indices**: text files listing mesh indices for train/validation/test.

2 Initial Inspection

In a `visualize.ipynb` notebook, I used `open3d` to:

- Load sample meshes.
- Confirm bounding boxes have max dimension 1.
- Observe that meshes were *not* centered at the origin.

3 Preprocessing Pipeline

For each mesh:

1. **Centering**: translate vertices so the mesh centroid is at the origin.
2. **One-ring graph**: extract adjacency lists of directly connected vertices.
3. **Geodesic graph**: run a Dijkstra-like expansion from each vertex up to distance 0.06 (since longest dimension = 1).
4. **Conversion to edge lists**: transform adjacency dicts into flat (i, j) edge lists required by GMEdgeConv.
5. **Packaging**: assemble a Python dict per mesh with keys:
   - `"obj_path"`: original file path
   - `"vertices"`: list of (x, y, z) coordinates

- `"one_ring"`: list of topological edges
- `"geodesic"`: list of geodesic edges
- `"joints"`: ground-truth joint coordinates

6. **Splitting**: save train, validation, and test sets to separate `.pkl` files based on provided index lists.

Performed in `preprocess.ipynb`

4 Visualization Check

Using `open3d.geometry.LineSet`, I rendered several edge lists over their meshes. The connectivity matched expectations for both one-ring and geodesic graphs, confirming correctness of preprocessing.