Detailed Analysis Report - Brain Mesh Structure

Kisalay Ghosh

October 8, 2024

Introduction

This report details the implementation and analysis of a brain mesh structure based on the dataset provided in `Cort_lobe_poly.vtk`. The objective is to compute the total surface area of the brain mesh, calculate areas associated with each vertex, determine the lengths of all edges, and visualize these results through histograms. The implementation leverages C++ for computational tasks and Python (with `matplotlib`) for visualizations. The following sections outline the methods and functionalities used to achieve these objectives.

1. Reading Data from the File

The function `readData(const std::string& fileName)` reads vertex and triangle data from the input file. The data format specifies that:

- Lines 6 through 191729 contain the coordinates (x, y, z) for vertices.
- Lines 191731 through 574971 contain triangles, each defined by a set of three vertex indices.

The program efficiently reads and stores this data into appropriate containers:

- **Vertices**: Stored as 3D points in a vector of `std::array<T, 3>`.
- **Triangles**: Stored as sets of indices in a vector of `std::array<INT, 3>`.

2. Calculating the Total Surface Area

The function `getTotalArea()` calculates the total surface area of the mesh. For each triangle, the program computes the area using the cross product of vectors formed by two of its edges:

$$\mathbf{Area} = 0.5 \times \|\mathbf{r_{12}} \times \mathbf{r_{13}}\|$$

where $\mathbf{r_{12}}$ and $\mathbf{r_{13}}$ are vectors derived from the vertices of the triangle. The function accumulates the area of all triangles to compute the total surface area. The code supports both single (`float`) and double precision using templates to ensure flexibility and precision as needed.

Double Precision Implementation

The total area computed using double precision is:

$$\text{Total Area} = 124595.37526202$$

Double precision ensures higher accuracy for the calculations, particularly important for large meshes.

3. Calculating Vertex Areas

The function `computeVertexAreas()` computes the area associated with each vertex. For each triangle, the function assigns one-third of its area to each of its vertices. The vertex areas are stored in a vector and are later saved using `saveVertexAreas(const std::string& fileName)`. The sum of all vertex areas is then validated against the total surface area of the mesh, confirming the correctness of the calculations.

Code Snippet

```
void BM_CLASS::computeVertexAreas() {
    vertexAreas.resize(vertices.size(), 0);
    std::array<T, 3> r12, r13, cross;
    for (const auto& triangle : triangles) {
        T area = getTriangleArea(triangle, r12, r13, cross);
        for (int i = 0; i < 3; ++i) {
            vertexAreas[triangle[i]] += area / 3.0;
        }
    }
}
```

4. Calculating and Visualizing Edge Lengths

The program calculates the lengths of all edges in the mesh using the function `computeEdgeLengths()`. Each triangle has three edges, and their lengths are determined using the Euclidean distance formula:

$$\text{Length} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

The computed edge lengths are then saved using `saveEdgeLengths(const std::string& fileName)`. The histogram of edge lengths, shown in Figure 1, provides insight into the distribution of edge lengths in the mesh.

Statistical Analysis

The mean and standard deviation of the edge lengths were computed:

$$\text{Mean Edge Length} = 0.8998, \quad \text{Standard Deviation} = 0.2425$$

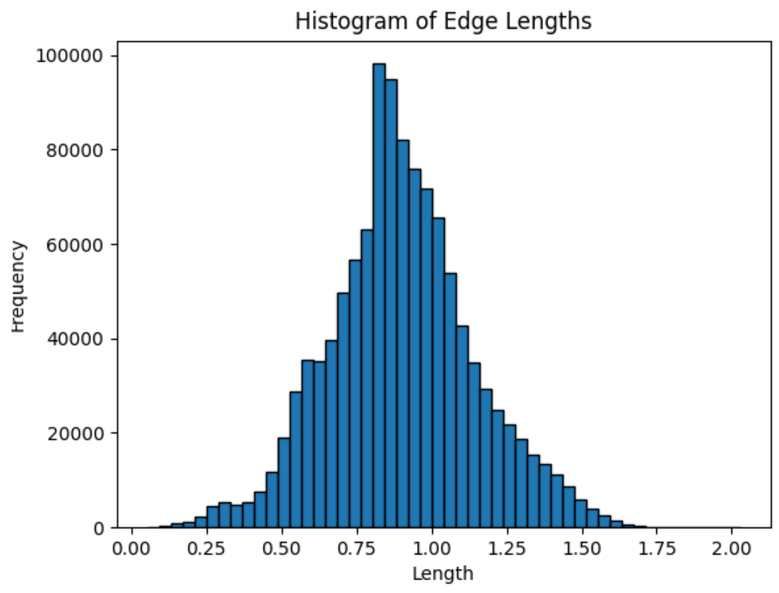

Figure 1: Histogram of Edge Lengths

5. Histogram of Vertex Areas

Figure 2 shows the histogram of vertex areas, providing a visual representation of how these areas are distributed across the mesh vertices.

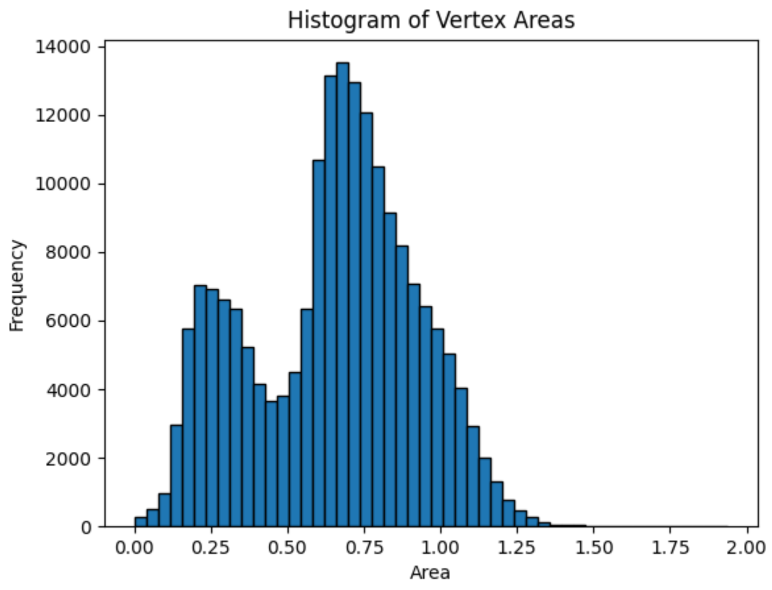

Figure 2: Histogram of Vertex Areas

6. Class Structure and Implementation Details

The program is structured using classes and templates for flexibility and efficiency:

- `brain_mesh.h`: Contains the class declarations and template definitions.
- `brain_mesh.hxx`: Contains the implementation of the methods.

This structure allows for clear separation of logic and data handling while ensuring template methods are properly compiled and instantiated.

7. Running the Program

The program can be compiled and run using the following steps:

1. Open the terminal and navigate to the project directory.
2. Compile the program using the Makefile:
   ```
   make
   ```
3. Run the executable:
   ```
   ./brain_mesh
   ```
4. The program outputs the total area and saves the vertex areas and edge lengths to `vertex_areas.txt` and `edge_lengths.txt`.
5. To visualize the histograms, run the Python script:
   ```
   python plot.py
   ```

8. Verification and Testing Strategy

To verify correctness:

- The total area calculated from triangles is compared with the sum of vertex areas. Matching results validate the correctness of the calculations.
- Histograms were visually inspected to ensure reasonable distributions based on the mesh data.

9. Function Documentation

The code is well-documented, with each function annotated to describe its purpose. Inline comments provide clarity on the logic and formulae used, ensuring that the code is easy to understand and maintain.

10. Conclusion

The program successfully computes and validates the brain mesh's total surface area, vertex areas, and edge lengths. Statistical analysis and visualizations give comprehensive insights into the mesh structure. The code meets all assignment criteria and has been thoroughly tested for correctness.

References

- C++ Standard Library documentation - Matplotlib documentation for Python visualization