CSED703M / AIGS703O: 3D Medical Image Processing and Analysis

Assignment #3

Deadline: Sunday, May 11th 23:59, 2025

## Objective

The objective of this assignment is to gain a fundamental understanding of mesh visualization and processing. It is important to note that the assignment mainly involves programming tasks, and thus it is recommended not to procrastinate until the deadline.

## Instructions (READ CAREFULLY)

- Complete individual steps and turn in your outputs (see **Task #**). There are a total of **3 Tasks** in this assignment.
- Paste (or screenshot) your corresponding codes to each task in your report. Your report should be self-contained. Please use a "**pdf**" format.
- Upload "**pdf**" (report) and "**zip**" (full source codes and vtk files used in your report) to PLMS before the deadline. You should include your name and student ID in your report.
- Please name the assignment files as "Student#.pdf" and "Student#.zip". Wrong file formatting will result in a **30% deduction**.
- Although you are free to discuss the problems with your classmates, you should complete the programming and writing tasks yourself. Submit all the source code you write along with detailed explanations of your approach to the problems.
- If you find yourself spending more than 24 hours on a single task, there may be an issue, or you may be overthinking. In this case, it is recommended to ask the instructor for advice to help you save time.

## 1. Mesh visualization

We learned basic functions in ParaView. Let's visualize the input mesh with its provided geometric feature (mean curvature). To read the input mesh file in MATLAB, use the following command:

```
[v_white, f_white] = read_vtk('lh.white.vtk');
```

Similarly, we can read the geometric feature file (lh.white.H.txt) using:

```
H = load('lh.white.H.txt');
```

To overlay the geometric feature (H) onto the input mesh, we can utilize the function write_vtk provided in the previous assignment as follows:

```
write_vtk('lh.white.H.vtk', v_white, f_white, struct('H', H));
```

We can use this function to add multiple features by simply adding fields in the struct. Please note that the first vertex index in read_vtk is 0 in f, and write_vtk assumes the first vertex index is 0.

- **Task 1-1**: Test the code above and visualize the input mesh with its geometric feature. Screenshot your visualization and include the output vtk files in your submission zip file. Do not forget to enable parallel projection to avoid any potential visual misinterpretation.

Sometimes, it can be challenging to visualize the buried regions in the input mesh. One way to improve visualization is to inflate the original mesh by updating its coordinates. To do this, run the following command in FreeSurfer:

```
mris_inflate ./lh.white.vtk ./lh.inflated.vtk
```

The resulting mesh will maintain the same connectivity as the input mesh, but with updated coordinates after the inflation process.

- **Task 1-2**: Repeat **Task 1-1** on the inflated mesh with its geometric feature (H) provided in **Task 1-1**. Screenshot your visualization and include the output vtk files in your submission zip file.

**2. Mesh smoothing**

Let's smooth the data we visualized in **Task 1**. We will use the Laplacian smoothing (1-ring neighborhood weighted average). Though there are several variations for this task, we will design data smoothing via linear algebra. Recall adjacency matrix we created in Assignment #2.

- **Task 2-1**: Make a sparse adjacency matrix of the original mesh (lh.white.vtk). To compute the length for each edge and its Gaussian weight, use the following steps: first, calculate the length for each edge. Then, determine the Gaussian weight for each edge with a mean of 0 mm and a standard deviation equal to the average of all edge lengths. Next, update the weights in the adjacency matrix accordingly and set 0 for the diagonal. Rescale each row so that its elements sum to 1. After constructing the sparse matrix denoted by A, data smoothing can be achieved quickly and efficiently.

- **Task 2-2**: To smooth the geometric data, perform matrix multiplication of A and H (mean curvature) and update H by this multiplication (i.e., H = A * H). Repeat this process for 10, 20, and 40 iterations. Then, visualize the results on the inflated surface and include the output vtk files in your submission zip file.

- **Task 2-3**: Perform **Task 2-2** again on the 3D coordinates of the original mesh. Then, visualize your results for 10, 20, and 40 iterations and include the output vtk files in your submission zip file. In your report, discuss how to overcome the shrinkage issue that can occur during the smoothing process. Please provide a narrative discussion without implementation.

**3. Re-tessellation**

We will create a new mesh using icosahedral surface. As studied, there are three main steps for re-tessellation: grid definition, closest triangle search, and barycentric interpolation. The icosahedral semi-uniform grids (icosahedral mesh files) are available at PLMS. In this assignment, we will focus on the closest triangle search and use a k-D tree here for the exercise; Do NOT use a different search method – in practice, this issue can be addressed by more efficient triangle search algorithms such as AABB. In MATLAB, we can construct a k-D tree using the following command:

```
MD = KDTreeSearcher(v);
```

To find the first to k-th closest points' IDs to a query point q, we can use the following command:

```
p = MD.knnsearch(q, 'k', k, 'distance', 'euclidean');
```

- **Task 3-1**: In some cases, the closest point to a query may not be a part of the enclosing triangle of the query point. Give an example case. Assume that all triangles and the query point are on the same plane.

To find the adjacent triangles of a given vertex, we can use a MATLAB built-in function.

```
tr = triangulation(f+1, v); % make sure the first vertex index is 1.
```

```
faces = tr.vertexAttachments(p); % this function will return face indices connected to vertex p.
```

The query points (i.e., the vertices of the icosahedral mesh) have the unit length. However, when testing whether a query point is enclosed by a given triangle, we need to relocate (rescale) the query point onto the plane of the triangle before the inside test. This is because the triangle is not curved and thus not on the sphere except at its three vertices (see Fig. 1). Therefore, we need to relocate (rescale) the query point onto the plane of ABC to ensure that the barycentric coefficients are calculated correctly. Note that this relocation should not be confused with orthogonal projection.

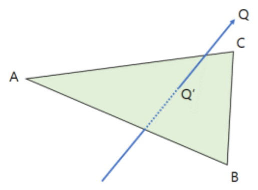

Fig 1. The plane of ABC is not a part of the sphere, which means that the query point Q needs to be relocated (rescaled) onto the plane of ABC to compute valid barycentric coefficients. This process is necessary because the triangle (plane) is not curved like the sphere except for its three vertices, and therefore using the original query point Q can lead to miscalculations of the barycentric coefficients.

- **Task 3-2**: Write Q' in terms of the normal vector N of ABC, Q, and A (or B or C), and illustrate your derivation.

Once Q' is ready, triangulation (MATLAB built-in function) computes barycentric coefficients:

tr.cartesianToBarycentric(faces, q_proj); % returns barycentric coeffs of input faces at query point (3D coordinate).

- **Task 3-3**: Implement closest triangle search and the associated barycentric coefficients. See the following pseudo code for your implementation guidance.

```
// initialization
Q = {q1 … qN};
k = 1;
// main loop
while Q != Ø
        for q in Q
                p = knnesarch(q, 'k', k);        // k-th nearest neighbors
                T = vertexAttachment(p);         // a set of triangles to p
                for t in T
                        q_proj = proj(q, t);  // from Task 3-2
                        bary = cartesianToBarycentric(t, q_proj);
                        if all coeffs in bary >= 0
                                // store this triangle as closest one to q
                                // store barycentric coeffs
                                Q = Q \ {q};
                        end if
                end for    // t in T
        end for    // q in Q
        k++;       // increase search range
end while
```

- **Task 3-4**: Compute the barycentric coefficients for the icosphere4.vtk, icosphere5.vtk, and icosphere6.vtk meshes. Use these coefficients to re-tessellate the original mesh and resample the geometric feature (mean curvature). Visualize your results for all icosahedral-tessellated "cortical" surfaces using ParaView and include the output vtk files in your submission zip file. To present your results in a clear and organized manner, create a 2-by-3 subplot as follows:
    - Top row: Show each surface with edges but without resampled geometric feature maps.
    - Bottom row: Show each surface without edges but with resampled geometric feature maps.
    - 1st column: Display the results for icosphere4.vtk

- 2nd column: Display the results for icosphere5.vtk
- 3rd column: Display the results for icosphere6.vtk.

**4. Spherical harmonics**

We can use spherical harmonic interpolation to reconstruct signals on the unit sphere. To generate the harmonic bases, we can use the following function available on PLMS:

Y = spharm_real(v, L);

The function generates a harmonic basis at degree L. We need to stack all bases into a single vector per vertex.

- **Task 4-1**: Call spharm_real(v, L) for L = 0 … 40. Concatenate all the bases to make a |v| by 41^2 matrix.

Once we create the basis matrix, we can find harmonic coefficients that fit the data in a least square manner. Once again, let's fit the 3D coordinates of the original mesh and the geometric feature.

- **Task 4-2**: Construct a linear system with harmonic bases. Use \ operator to solve the system in MATLAB.

Now, we want to reconstruct signals using icosahedral mesh.

- **Task 4-3**: Compute the new bases from icosphere4.vtk, icosphere5.vtk, and icosphere6.vtk. Reconstruct signals using the coefficients computed in **Task 4-2**. Like **Task 3-4**, visualize two rows for the 3D coordinates and geometric feature (mean curvature) on the reconstructed icosahedral "cortical" surfaces, respectively, at degree of 10, 20, and 40. Also, include the output vtk files in your submission zip file.
- **Task 4-4**: Discuss advantages and disadvantages of harmonic-based re-tessellation and search-based approach.