AI601/CSE613: 3D Medical Image Processing and Analysis

Assignment #2

Due on Friday, April 7th, 2022, 11:59:59 pm

Objective

The goal of this assignment is to understand **surface reconstruction** using simple processing techniques. We will use some of functionality written in MATLAB in FreeSurfer (volumetric image reader, etc.). Note that you will be mostly asked for programming in this assignment, so please do not wait until the last minute.

Instructions

- Complete individual steps and turn in your outputs (see **Task #**). There are a total of **8 Tasks** in this assignment.
- Paste (or screenshot) your 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) to BlackBoard before the deadline.
- Because FreeSurfer uses a random seed, the processed data in your former assignment might slightly differ from what your classmates generated. Use "T1.nii.gz" and "ribbon.mgz" in BlackBoard.

1. Brain mask

Bran tissue segmentation is involved with a series of complex processing techniques. In this task, we will use the processed data by FreeSurfer, which is a part of "recon-all". Due to the randomness in FreeSurfer, you are asked to use the provided volume (ribbon.mgz) rather than what you processed. The volume file contains the processed image with white and gray matter tissues in both left and right hemispheres. Each region is encoded with distinct integer numbers. Our first task is to generate brain mask.

- **Task 1-1**: open and browse "ribbon.mgz" in Freeview. List all integers and summarize which number belongs to which tissue type in which hemisphere (LH-WM (left hemisphere-white matter), LH-GM. RH-WM, RH-GM). Fill in a table to summarize your answer.

	LH-WM	LH-GM	RH-WM	RH-GM
Label (int)				

  Hint: click voxels to see their labels. Also, find "L" and "R" in your Freeview display to determine left and right hemispheres. If you switch the color map to "Lookup Table", you can also get the hemisphere information.

To read the volume data, let's use "MRIread" in MATLAB. This is FreeSurfer's implementation, so if your MATLAB environment cannot recognize the function, add its path ($FREESURFER_HOME/matlab). You can read in your data with "mri = MRIread('ribbon.mgz');". The function returns a structure type. The volume information is stored in "mri.vol" as a regular matrix.

Let's make a binary mask for all combinations in **Task 1-1**. For the white matter (WM) masks, extract a binary mask from each hemisphere. For the gray matter (GM) masks, combine the "gray and white matter tissues" to create **a filled mask for each tissue in each hemisphere**, resulting in a total of four filled masks. These masks

will be used in **Task 2**. Once completed, overlay your masks on the original T1 image (T1.nii.gz). Here is an example I/O code for volume data in MATLAB.

```
mri = MRIread('ribbon.mgz');

mri.vol = … % do some operations to combine tissues

MRIwrite(mri,'./mask.nii.gz','uchar');
```

Note: do NOT apply filtering such as morphological operations on the masks as it can alter them unexpectedly.

- **Task 1-2**: screenshot four of your binary masks in a total.

Note: do NOT make Boolean outputs but use an integer range, [0, N] (N > 0), instead. N can be any number.

2. Simple surface reconstruction

Iso-surface is one of the discrete representations of surfaces. In this task, we want to realize level-sets as triangle mesh structures using the binary masks that we created in **Task 1-2**. Here, we can think the binary masks as a continuous function (0 to N).

To generate an iso-surface for each binary mask, we will use MATLAB implementation. In this implementation, the iso-surface is reconstructed by finding the boundary of a given level. Therefore, you need to determine the appropriate level (e.g., N/2), and then the "isosurface" function can trace along the boundary of that level. Here is an example code for realizing the iso-surface.

```
[f,v]=isosurface(vol, 0.5);
```

Since the surfaces are created from the volume matrix, we need to provide a valid transformation to update vertex coordinates. Otherwise, the resulting coordinate system will be based on the volume indices. To this end, we can apply a simple matrix transformation.

```
n = size(v, 1);

v = [v-1, ones(n, 1)] * mri.vox2ras';

v = v - [mri.c_r, mri.c_a, mri.c_s, 0];

v = v(:, 1:3);
```

To store results as a VTK format, use the provided MATLAB function available at Blackboard.

```
write_vtk('output_filename.vtk', v, f-1);
```

It's that simple. You can then load your surface in either Freeview or ParaView (surface only). For volume-surface overlay, let's use Freeview. So, you need to do:

- **Task 2**: screenshot volume-surface overlay (the original T1 and reconstructed surfaces) like what you have already done in assignment #1.

Note that when loading data, you should always open the volume data first, and then load the surfaces. This is because FreeSurfer's surface outputs contain the original coordinate system of the MR scans, but generic mesh files like VTK do not include such information. If you notice that the reconstructed surfaces and the volume do not match well, double-check the order in which you loaded the data.

3. Graph representation

Reconstructed mesh structures often suffer from reconstruction errors, such as noise and discretization artifacts. To clean up isolated structures, we will create mesh connectivity using a sparse adjacency matrix. To do this, you need to first compute a full list of edges. In your iso-surfaces, the face information is stored in the "f" variable.

- **Task 3-1**: from f, make a 2D list (2 columns) of edges with a pair of directions to make sure your list encodes an undirected graph. For face (v1,v2,v3), your list should contain { (v1,v2), (v2,v3), (v3,v1), (v2,v1), (v3,v2), (v1,v3) }.
- **Task 3-2**: create a sparse matrix from your edge list in **Task 3-1**. You may want to use the "sparse" function in MATLAB; i.e., "sparse(from, to, value, size, size);". To create a binary matrix, set value = 1.

4. Mesh cleanup

Next, we need to compute the connected components from each surface to eliminate isolated chunks. A sparse adjacency matrix (such as A) has the advantage of easy conversion into a graph form. You can achieve this by calling the MATLAB function "graph": "G = graph(A);". This graph structure has many useful supporting functions. We will use "[bins, binsizes] = conncomp(G)" to find connected components. You can then check the number of vertices that belong to each component in "binsizes", as well as the component labels in "bins".

We need to create a function that can (a) take a component label (say, 1) and (b) generate a new mesh with that label. To do this, you need to extract all the vertices with label 1, and then reorder (squeeze) vertex IDs as vertices with other labels will be removed from the original list of vertices. For example, if you remove one triangle (the first three vertices), you need to update them as follows.

1 2 3 (label 2) → deleted

4 5 6 (label 1) → 1 2 3

7 8 9 (label 1) → 4 5 6

Once unnecessary vertices are cleaned up and face information is up to date, you can write a new mesh file using "write_vtk" in MATLAB.

- **Task 4-1:** find the largest component of your mesh structures. Like **Task 2**, overlay your new surfaces onto the original T1 image. Explain your logic in your report. **Hint:** see "ismember" function for efficient processing.

Let's investigate your data further. Once you create a sparse adjacency matrix of the largest component, it is easy to count how many edges exist in that mesh. Let's explore a useful property.

- **Task 4-2**: compute the Euler characteristic of your mesh. Explain your data in terms of genus. How many holes in your data based on genus #? Also state # of vertices, faces, and edges to show how you obtain genus #.

Finally, we want to understand the isolated regions.

- **Task 4-3**: find the second largest component of your mesh structures in LH-GM. Overlay that component onto the original T1 as well as the tissue mask you created in **Task 1-1**. Discuss why such a structure appears and how we can remove during the mask creation. What would be advantages and disadvantages in your approach? Justify your answers.