Averaging Algorithm Writeup

Rahul Ramesh and Trent Lau

January 3, 2021

1 Specifications

The input is a set of STL mesh files. It is expected that each object is oriented in a consistent direction, though they need not be centered.

The output is an STL mesh file that reflects the average composition of each of the input files.

2 Algorithm Design

Step 1: Each mesh is loaded into the program as a trimesh with the python Trimesh package. Trimeshes store objects as a set of vertices and a list of triplets of vertices connected by faces.

Step 2: If given in millimeters, the input mesh's units are converted to inches. This transformation is currently applied to all inputs, so the algorithm is not coded to mix and match inputs in different units.

Step 3: Each object is then centered, so that the centroid of the object lies at the origin of the coordinate plane. There are two types of centroids explored: a centroid derived from the object mass itself and a centroid derived from the bounding box enclosing the object. More information can be found in the Tuned Parameters section.

Step 4: The Signed Distance Function (SDF) of each mesh is calculated at a resolution of 100 pixels. As a result, at 100x100x100 points on the 3d object, the vector (which encodes the distance and direction) from that point to the nearest surface point of the mesh is stored. The pixel resolution another tunable parameter.

Step 5: The resultant SDF voxels are averaged together, coordinate by coordinate. If the user provides weights for each input mesh, a weighted average can be calculated, though by default the arithmetic mean is used.

Step 6: The Marching Cubes algorithm is applied from the Skimage package to reconstruct a mesh from the given averaged voxel.

Step 7: The resultant mesh is healed by removing any extraneous surfaces created by Marching Cubes. Only the largest contiguous surface is kept.

Step 8: Since the Marching Cubes output is not scaled properly, it must be scaled to match the input sizes. The average mesh is scaled to meet the average size of the inputs, as detailed in the ”Final Scaling” section of tuned parameters.

Step 9: The final average mesh is then exported as an .stl file.

3 Tuned Parameters

3.1 Centroid

Input samples are not presumed to be aligned. To account for this, each input mesh is centered upon loading in by subtracting the centroid of that object from each vertex coordinate. However, there are two distinct ways to calculate the centroid of an object, and thus two distinct ways of centering meshes. One method is calculating the exact center of the bounding box of the object: the smallest rectangular prism that fully encloses the object. The other method is by directly calculating the centroid of the object using the Trimesh package. Though the two do not differ significantly, the direct centroid respects the variable density of the original object while the bounding box centroid respects the extremities.

3.2 Final Scaling

The object created from the skimage implementation of marching cubes was not properly scaled to the size of the inputs. Therefore, scaling of the average had to be done manually. The mesh average was thus scaled to the average sizes of each input, equally weighted unless otherwise denoted.

Two metrics were used to establish the sizes of the inputs. The first metric was volume; the average mesh was scaled to have volume equal to the average volume of the inputs. The second metric was the normalized distance of a mesh, or the Euclidean distance from the center of the mesh to the farthest vertex. Both metrics worked, but the volume one is preferred as the normalized metric tended to create slightly larger average meshes than was expected.

3.3 Marching Cubes Resolution

This value is the resolution of the voxels calculated from each input mesh. It is tuned to be 100, since higher values appeared to crash the program by forcing it to hold too much information in memory. We also experimented with a resolution of 64.

4 Visualizations

Figure 1: A collection of Atlas bones (right) and the resultant average created by the algorithm (left)

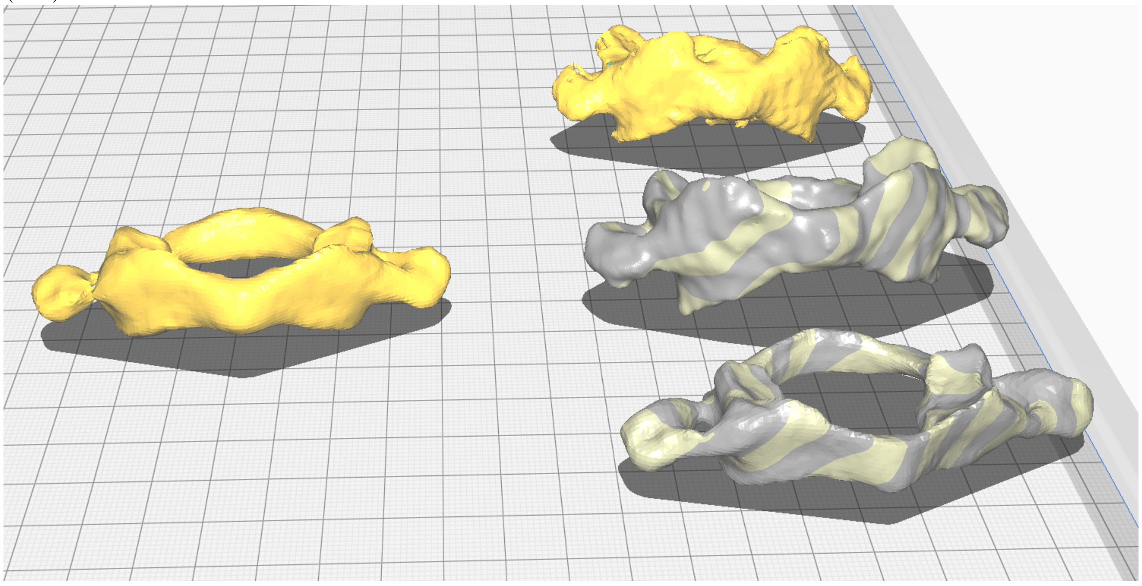


Figure 2: Heel bones (right) and average (left)

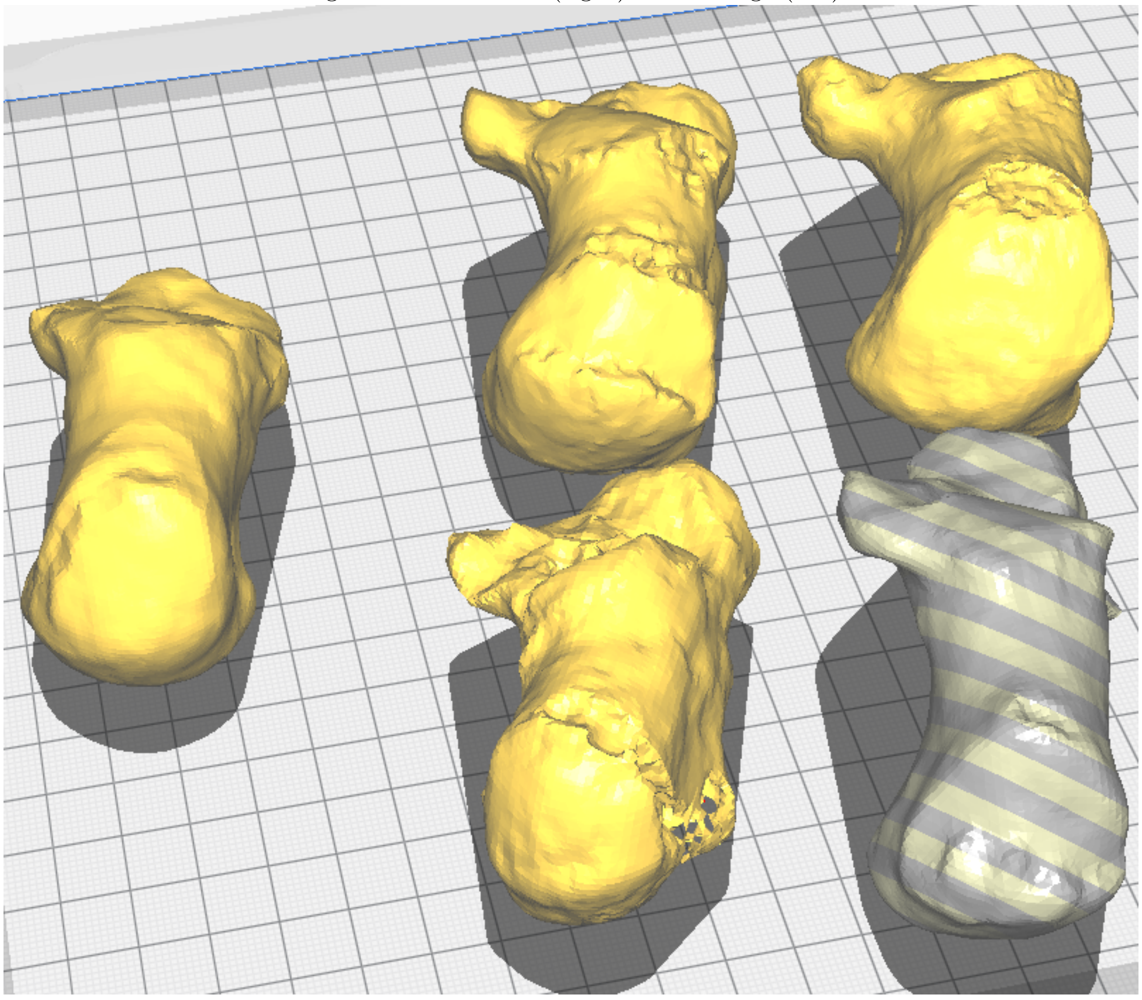
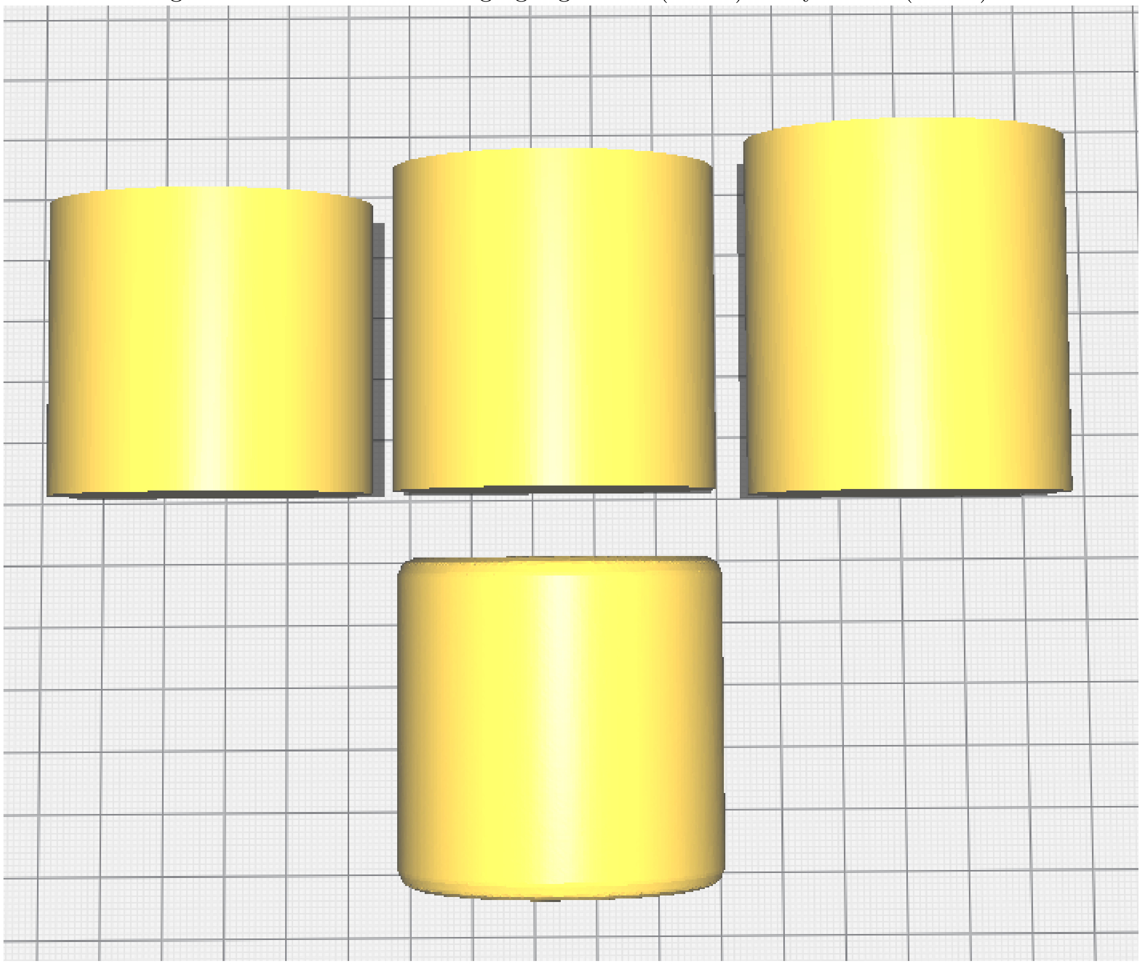


Figure 3: A test of the averaging algorithm (below) on cylinders (above).



5 Implementation Details

5.1 coding

Python 3 was used with an anaconda distribution.

5.2 imports

- Trimesh (storing and manipulating meshes)
- skimage (implementation of marching cubes)
- numpy (general vector/matrix arithmetic)
- mesh-to-sdf (implementation of calculating the SDF voxel), used with permission and found here: https://github.com/marian42/mesh_to_sdf