Geometric Data Processing

Assignment 1: Mesh and Surface Analysis and Surface Registration

Hendy Liang - 5027268, Andrej Tibenský - 5882133, Yimin Zhou - 5881870

May 17, 2023

1 Algorithms and Functionality Implemented

1.1 Analysis

1.1.1 Genus of the Surface

We implemented the algorithm for genus calculation for models that have no boundary loops, and a function allows the user to do genus calculation on chosen model. Shown in Fig 1.

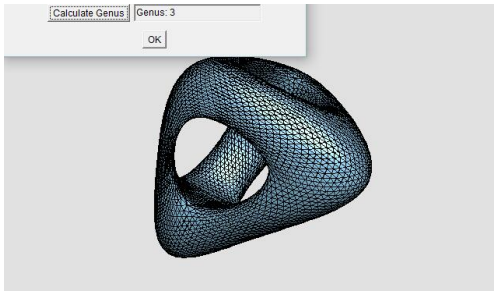

Figure 1: 3 genus mesh.

1.1.2 Volume Enclosed by the Surface

The algorithm for volume calculation for models that are closed, and a function allows the user to do volume calculation on chosen model. Shown in Fig 2. Initially, the function checks if the mesh is closed, if not, it returns a warning. If the mesh is closed, it iterates over each element (presumably a triangular face), calculates the cross product of two vertices, and then performs a dot product with the third vertex, scaling the result by 1/6 (we use absolute value here, due to vertex order different from models). The sum of these values over all elements yields the volume.

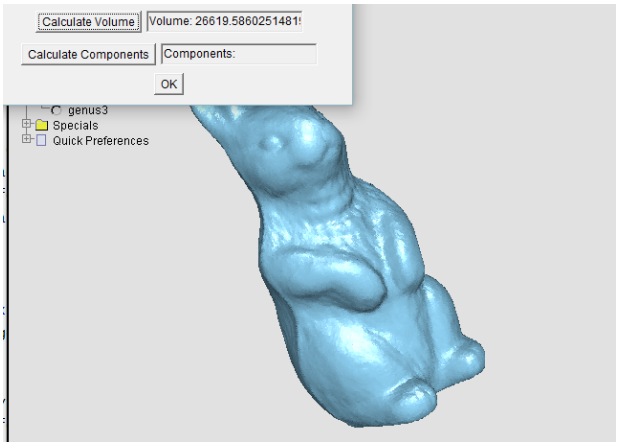

Figure 2: a closed model.

1.1.3 Number Connected Components of the Mesh

The algorithm calculates the number of connected components in a model, shown in Fig 3. It works by maintaining a queue to perform a breadth-first search (BFS) on the elements of the mesh. Each element is checked to see if it has been visited. If it hasn't, the algorithm counts it as a new component and then iterates over all its neighbours, marking them as visited and adding them to the queue. This process continues until all elements have been visited. The total number of components is returned.

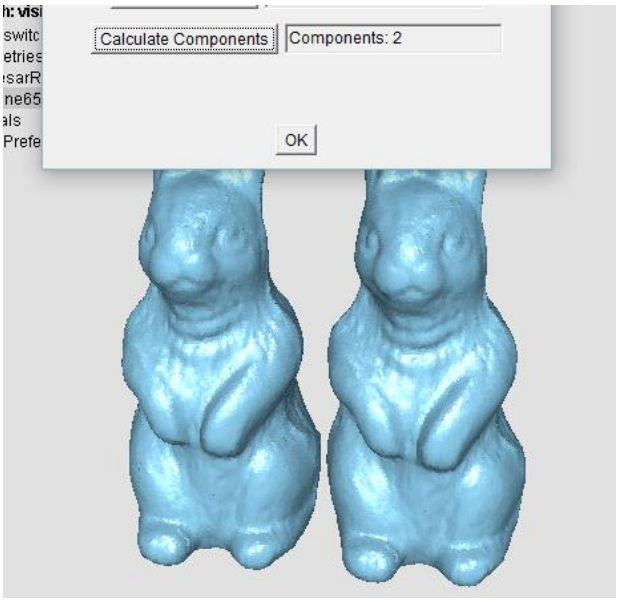

Figure 3: a model with 2 components.

1.2 Rigid Registration

For rigid transformation, we iterate the following steps, until reach the user's set max iteration number. The initial state is in Figure 4, and the icp result is shown in Figure 5.

1.2.1 For Every pi Find the Closest Vertex qi in Q

We have 2 methods, one is point-to-point the other is point-to-plane. The findClosestVertex function performs a brute-force search to find the vertex closest to a given point. It iterates over all vertices, calculating the distance to the point, and retains the one with the smallest distance. The findClosestPlane function similarly finds the closest plane to a given point. It calculates the plane's normal vector, computes the distance from the point to the plane, and retains the plane with the smallest distance, projecting the point onto this plane.

1.2.2 Compute the Median Distance

We calculate the median distance from a sorted list of distances. It determines if the total number of distances is even or odd, and calculates the median accordingly. Then, it sets a threshold by multiplying the median distance by a factor k. Following this, it generates new lists of vertices, by retaining only those vertices whose corresponding distances are less than or equal to the threshold. Finally, it replaces the original vertex lists with the newly created lists.

1.2.3 Compute the Optimal Rigid Transformation Matrix

We calculate the optimal rigid transformation of a set of vertices. It first calculates the centroids of two sets of vertices, p and q, by summing all vertices and dividing by the number of vertices. Then, it constructs a matrix M based on the outer product of the differences between each vertex and the respective centroid. SVD of M is computed, and the determinant of the multiplication of V and U transposed (from the SVD) is used to create a correction matrix. This correction matrix helps to compute an optimal rotation matrix R. A translation vector t is then calculated to align the centroids. Finally, the optimal rigid transformation (rotation and translation) is applied to all vertices in set P, and the transformed set P is updated for display.

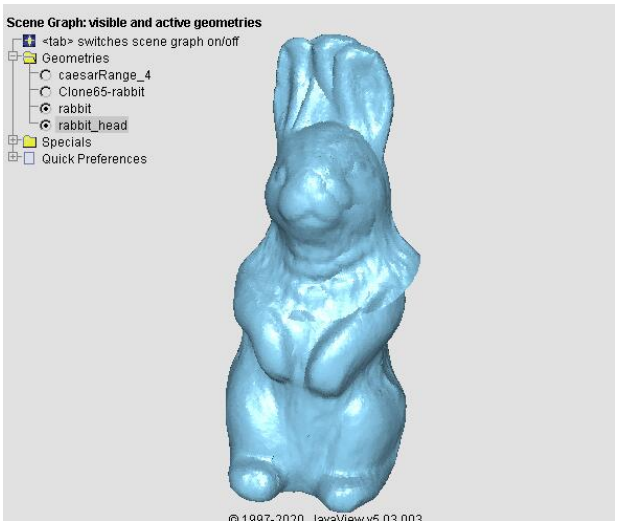

Figure 4: Two models with different positions.

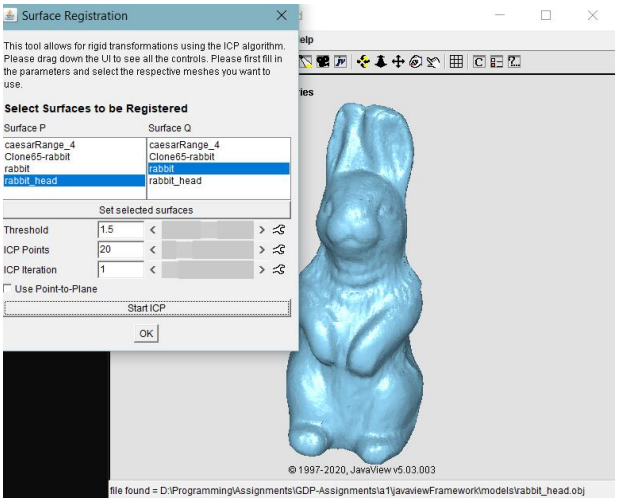

Figure 5: The head transformed to the Rabbit head position.

2 How the implementation can be used

We implemented the tasks as workshops inside JavaView 5.03.

2.1 Geometry Analysis

The mesh analysis is inside the MyWorkshop workshop and gives information about the most recently imported geometry. At the top of the window, the number of triangles is shown followed by controls for the X Offset of the mesh. Finally, the window provides 3 buttons for calculating the genus, volume and components of the most recently imported geometry.

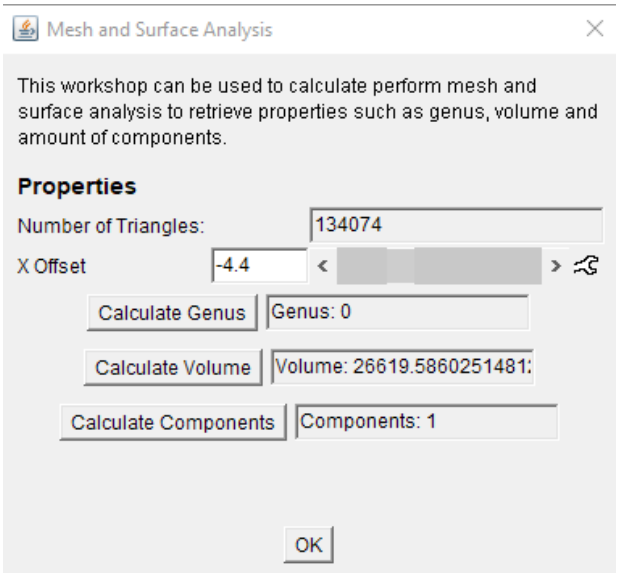

Figure 6: The geometry analysis window.

2.2 Surface Registration

The surface registration, using the Iterative closest point algorithm, is provided inside the Surface Registration workshop. First, we allow the user to select a surface P and surface Q from the currently imported geometries. Next we let the user choose the distance threshold, the number of ICP Points, the number of ICP iterations and whether to use Point-to-Plane instead of Point-to-Point. Finally, we provide a button to run the ICP algorithm on the selected geometries using the provided parameters. After the ICP is run, the window shows the MSE loss between the two geometries.

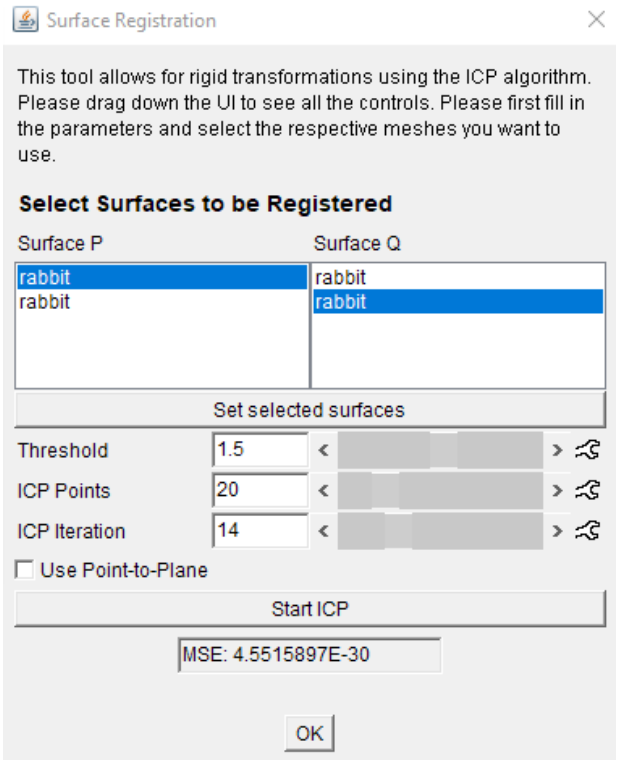

Figure 7: The surface registration window.

3 Verification

For the genus and amount of connected components, we simply manually inspected the results and the mesh itself. These should all be relatively easy to spot with the blind eye. As for the volume, it is a bit harder to verify it since we do not know the ground truth. However, we can compare meshes to each other to see if one mesh is bigger or smaller than the other mesh.

As for the iterative closest point algorithm, we can verify it by testing on two meshes and seeing how one moves in relation to the other. The best example for this is looking at the partial bunny head and the full bunny, the partial head should move to the position in the full mesh and it does.

4 Evaluation

On default, we use a threshold of 1.5, 20 points and 1 iteration. The user is free to adjust these to whatever they want. We have added a MSE calculation to show the difference between the new transformed vertex and the target vertex.

Changing the number of iterations determines how far the algorithm has converged. If it is already converged, it will stay in the proximity of the local minimum it has found.

Raising the threshold allows for more exploration, but also means the algorithm might go into an area that is worse than the previous iteration.

The amount of points determine the accuracy of the transformation to some degree, since more vertices are considered, the more information we are using. This comes at the cost of computation, since we are using a brute force search method to compute the median distances, this could become very expensive when the amount of points is large.