Critical Review: A Comparison of Mesh Simplification Algorithms

Nicole Ortega, Group 16

Abstract

The goal of our project is to develop a browser-based constructive solid geometry application for 3D anatomical models. "A Comparison of Mesh Simplification Algorithms" [1] by Cignoni, Montani and Scopigno can be best described as a literary review paper that characterizes fundamental simplification methods and compares six simplification methods using the Metro Tool. Ultimately, knowledge derived from this paper, as well as other cited sources, will be applied to determine which simplification method will perform best on anatomical models for use in a bowser-based environment.

Project Overview

One in 323 children in the US are born with cerebral palsy. Two out of every three children born with cerebral palsy could walk if they had proper orthotics to alleviate their condition. Fusiform, a medical device company has developed a process to reduce waste, reduce time and increase efficiency of orthotic design and fabrication. Currently, the process demands approximately 10 hours to create a single custom design orthotic in SolidWorks. The goal of our project is to develop a browser-based constructive solid geometry application for 3D anatomical models. This application would allow the orthotist to add pre-designed orthotic components onto a 3D leg scan of cerebral palsy patients.

Simplification Methods and Characteristics

Complex meshes are great for precise representations of a model. However, ……..There are many different methods that are implemented for simplification of a mesh. Among them include, but are not limited, to the methods discussed below.

- Coplanar facets merging – multiple triangles in the same plane merge into a large polygon. The polygon is then divided into fewer triangles, called re-triangulation.
- Controlled vertex/edge/face decimation – vertices/edges/faces that fit a specific criterion are removed. In the case of vertex and edge decimation, the resulting hole is re-triangulated.
- Re-tilling – random vertices are inserted into the mesh and moved to areas of curvature. This is then followed by vertex decimation.
- Energy function optimization – an energy function is defined to measure the quality of the mesh. Actions (collapsing, swapping, or splitting) that fit a specific criterion and cause the lowest increase to the energy function are executed.
- Vertex clustering – clusters of vertices are

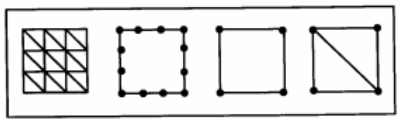

Figure 1. Coplanar facets merging

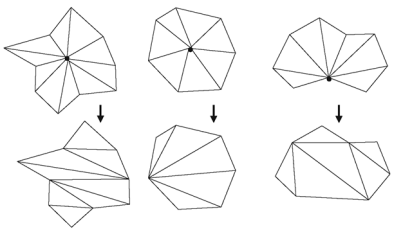

Figure 2.Re-triangulation

combined into one vertex with its positioned usually determined by the average position of all the vertices.

Simplification algorithms use one these methods, or modified implementations, to perform simplification. To avoid describing how each algorithm performs simplification, the paper characterizes algorithms (on Table 1[1]) with the following features:

- Optimization goal – minimize the size of the mesh given an error or minimize the error given a size
- Incremental simplification – multiple iterations of the algorithm
- Topological features – feature which the algorithm modifies. This includes vertices, edges, faces, and vertex pairs
- Local/global error (or other) – approximation error calculated on a local level (specific area), on a global level (entire mesh), or by other means (energy func)
- Bounded error – error restricted to an upper and lower bound
- Preservation of global mesh topology
- Relocation of vertices
- Preservation of solid/features edges or angles
- Multi-resolution output
- Speed and availability

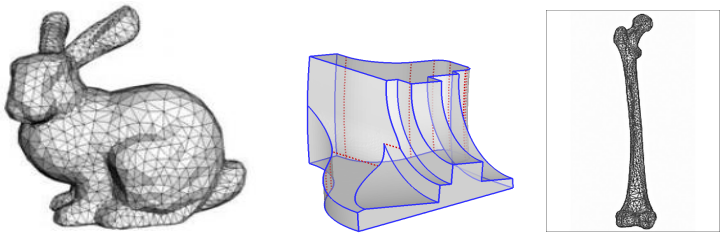

Figure 3. From left to right: Bunny, Fandisk, and Femur

To compare mesh simplification algorithms, six simplification algorithms were tested on three meshes: the bunny, the fandisk and the femur. The performance of each simplification algorithm was evaluated using the Metro Tool. The Metro Tool is a tool to evaluate approximation decision. The tool allows for updates at different levels of detail (number of vertices and triangles) and does not require knowledge of what method was used for the simplification. The approximation error is used to evaluate differences in precision. To compute the approximation error, the simplified mesh is sampled and a point-to-surface distance to the original mesh is computed. The outputs to compare likeness are as follows: $N_{vertices}$, $N_{triangles}$, E_{max}, E_{avg} ,Time, Edge, Length, Area, and Memory (kb). This can be seen on Tables 2, 3, and 4 on the paper.

Simplification Algorithms

Six simplification codes were tested on the following three meshes: bunny, fandisk and femur. The simplification algorithms are characterized and briefly described below, as well as detailed in cited sources in the paper.

Mesh Decimation [2] Characterization: Decimation algorithm, minimize error, incremental, vertices, local error, no bounding error, preservation of mesh topology, no relocation of vertices.

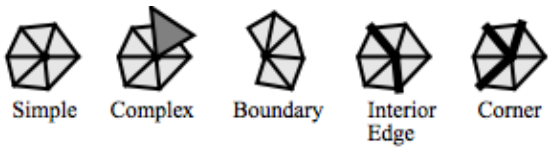

Figure 4. Characterization of vertices [2]

Mesh Decimation begins by characterizing the local vertex geometry and topology into one of five options:

simple, complex, boundary, interior edge, and corner. Complex vertices are not deleted. Each vertex is then evaluated using decimation criteria. The decimation criterion calculates the vertex distance to the average plane for simple vertices or the vertex distance to the edge for boundary and interior vertices. If the vertex distance is less than *d* then the vertices (and all of its associated triangles) are deleted. The resulting hole is re-triangulated and the process is repeated.

Multi-resolution Decimation [3] Characterization: Decimation algorithm, minimize error and size, incremental, vertices, global and bounded error, multi-resolution output, preservation of mesh topology, no relocation of vertices.
Multi-resolution Decimation uses JADE (Just Another Decimator), an enhanced decimation algorithm. JADE uses the same classification of vertex topology as the Mesh Decimation algorithm (Figure 4). It then evaluates each vertex using a global approximation error criterion. The vertices with minimal local and accumulated global errors are selected for deletion. The resulting hole is re-triangulated using edge flipping and the process is repeated.

Simplification Envelopes [4] Characterization: Decimation algorithm, minimize size, not incremental, vertices, global and bounded error, preservation of mesh topology, no relocation of vertices.

Simplification Envelopes surround the mesh with an inner and outer mesh, an envelope. The envelope is a user specific distance *e* from the mesh. Vertex decimation occurs, but the resulting simplified mesh must stay within the bounds of the envelope. This enables preservation of global topology.

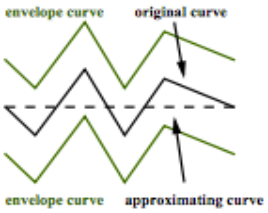

Figure 5. Simplification Envelope [4]

Mesh Optimization [5] Characterization: Energy Optimization algorithm, minimize size, incremental, vertices and edges, other error, no bounded error, preservation of mesh topology, relocation of vertices.
Mesh Optimization minimizes the following energy function: $E(K, V) = E_{dist}(K, V) + E_{rep}(K) + E_{spring}(K, V)$. The first term describes the distance energy, the sum of the squared distances of the points from the mesh. The second term embodies the representation energy, which penalizes meshes with large number of vertices. The last term is the spring energy, which acts like placing a spring on the edge of the mesh with rest length 0 and tension k. This term is used to regulate optimization to a desired local minimum.

Progressive Meshes [6] Characterization: Energy Optimization algorithm, minimize size, incremental, edges, other error, no bounded error, multi-resolution output, preservation of mesh topology, relocation of vertices.
Progressive Meshes is similar to Mesh Optimization. The following equation describes the energy function: $E(M) = E_{dist}(M) + E_{spring}(M) + E_{scalar}(M) + E_{disc}(M)$. This equation includes the distance and spring energy terms found in Mesh Optimization, but also includes two other terms that preserve attributes of the mesh. The third term represents scalar energy, which measures the accuracy of its scalar attributes such as diffuse color, normal, texture coordinates

and shading parameters. The fourth term is the disc energy, which measures the geometric accuracy of discontinuity curves (ie. sharp edges).

Quadratic Error Metrics Simplification [7] Characterization: Clustering algorithm, minimize size and error, incremental, vertex pairs, global error, not bounded error, multi-resolution output, no preservation of mesh topology, relocation of vertices.
Quadratic Error Metrics Simplification computes the Q matrix for all the initial vertices on the mesh. The Q matrix is the quadric error, sum of fundamental error quadrics. Quadrics are created by the planes that meet at the vertex. The quadric error is calculated using the formula $\Delta v = v^T \mathbf{Q} v$. Valid vertex pairs are chosen if the pair is an edge or the Euclidian distance is less than a threshold. The optimal contraction of a valid pair is determined by the equation $v'^T(Q_1 + Q_2)v'$ and is designated as the cost of the contraction. Vertex pairs with minimal cost are contracted and the whole process is repeated.

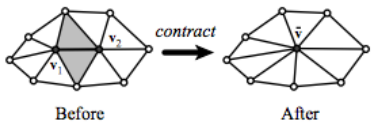

Figure 6. Vertex contraction [7]

Results
After running tests, the authors found that Mesh Decimation and Simplification Error failed to reach high simplification rates on the femur data. They suspect that the problem arises from both algorithms removing vertices in random order. The authors suggest that iterating multiple times through the mesh would provide a partial solution. Additionally, they found that Progressive Meshes and Mesh Optimization provide the best average error across the meshes. Simplification Envelope and Multi-resolution Decimation provided the best results when high accuracy was needed. Interesting results were found for the Quadric Error algorithm. When the algorithm was used for the fandisk mesh, fast speeds and small errors were attained. However, large errors were attained for the femur and bunny mesh. The authors believe that these large errors occurred because the meshes had open boundaries. The authors claim that inserting perpendicular planes at boundary edges and assigning large costs to deletion of these vertices can reduce the error.

Project Analysis: Femur Data
Since our project uses anatomical models, it is important to closely analyze the results of the femur mesh. On average, our 3D leg scans contain 47 thousand vertices and 92 thousands triangles. In comparison, the femur mesh contains 76 thousand vertices and 153 triangles. In order for the application to update at a decent speed, we need to simplify our mesh down to 10 thousand vertices. We also need our mesh to load onto the browser in less than a minute. Preservation of mesh topology and features is important in this medical application to build a custom orthotic from the 3D leg scan. Thus, I compared the results for each method on the femur data at 10% simplification (approximately 8 thousand vertices). N_{vert} and Area were not considered because of their similar results across algorithms. Edge length was not considered because of large variability within a single algorithm. The following results were found, with 1 denoting the best performance and 5 denoting the worst performance.

	E_{max}	E_{avg}	Time	Mem. kb

Mesh Decimation	5	6	1	1
Simplification Env.	1 (Best)	5	4	4
Multiresolution Dec.	2	3	3	2
Mesh Optimization	4	1	6	3
Progressive mesh	3	2	5	N/A
Quadric Error Metric	6 (Worst)	4	2	N/A

In summary,

- Simplification Envelope performed best in terms of smallest max error
- Mesh Optimization performed best in terms of average error
- Mesh Decimation performed the simplification the fastest
- Mesh Decimation reduced the size of the mesh the most

Applications

In order decide which algorithm to apply to our 3D leg scan, preservation of mesh, accuracy, and speed must be considered. Immediately, Quadric Error Metric is eliminated. Not only does it have the worst maximum error, but also its use of the clustering method does not preserve mesh topology. In terms of accuracy, all methods have similar average errors. Speed, on the other hand, differs greatly across algorithms. Because we are trying to create an application that works on a browser, speed is our primary concern. With this in mind, I believe that Mesh Decimation would produce the best results. Additionally, I believe that if a bounded error criterion (similar to Simplification Envelope) were created and implemented using a kd tree, the maximum error would be reduced. However, this might come at a run-time cost.

Paper Critiques

Looking positively, the paper was thorough, had a good overview of methods, contained complete and detailed tables and covered a wide range of methods with its simplification codes. However, the paper also had a lot of negative aspects. The content was too broad and needed a clear and specific focus. The tables, although informative, were cluttered, hard to read, and at times confusing. The choices of characterization features were not explained (except for mesh topology). Additionally, it would have been helpful to summary the methods used in the six simplification codes and provide a table summary of the results. Lastly, and most importantly, the results lacked deep analysis and mainly appeared to regurgitate information seen in the tables.

Citations

[1] Cignoni, P., C. Montani, and R. Scopigno. "A Comparison of Mesh Simplification Algorithms." *Computers & Graphics* 22.1 (1998): 37-54. Web.

[2] Schroeder, William J., Jonathan A. Zarge, and William E. Lorensen. "Decimation of Triangle Meshes." *ACM SIGGRAPH Computer Graphics SIGGRAPH Comput. Graph.* 26.2 (1992): 65-70. Web.

[3] Ciampalini, A., P. Cignoni, C. Montani, and R. Scopigno. "Multiresolution Decimation Based on Global Error." *The Visual Computer* 13.5 (1997): 228-46. Web.

[4] Cohen, Jonathan, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. "Simplification Envelopes." *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '96* (1996). Web.

[5] Hoppe, Hugues, Tony Derose, Tom Duchamp, John Mcdonald, and Werner Stuetzle. "Mesh Optimization." *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '93* (1993). Web.

[6]Hoppe, H., Progressive meshes. In ACM Computer Graphics Proc.. Annual Conference Series (Siggraph '96), 1996, pp. 99±108.

[7] Garland, Michael, and Paul S. Heckbert. "Surface Simplification Using Quadric Error Metrics." *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '97* (1997). Web.