

Mesh Simplification

Independent Study Project Report

by
Akash Kushal
kushal@uiuc.edu

under the guidance of
Prof. Michael Garland

1 Introduction

The models produced by high-precision laser scanners or automatically generated polygonal surfaces are often very densely tessellated. This detail is not required for various application such as interactive display. Hence, there is a need to reduce to complexity of the model but at the same time keep the geometry of the object being represented more or less the same. The current techniques for simplification can be categorized into 2 broad classes. One class contains the iterative edge contraction based algorithms which work by iterative picking an edge on the mesh, collapsing both its endpoints into a single vertex and continuing in this way until the size of the model reaches the desired size. But, these schemes are not practical for very huge models that do not fit into the main memory. The algorithm in [1] falls into this class. The second class of algorithms based on vertex clustering work by first discretizing the mesh into a grid of voxels. They may later process this discretized model in different ways. Some recent algorithms in this class include the ones proposed in [2] and [3]. These algorithms produce results of a lesser quality than the iterative edge contraction schemes but are the only viable options for large meshes.

2 Our Work

As noted above for compressing very large meshes the iterative edge contraction algorithms become impractical. We would need an algorithm that does some number of linear passes on the mesh data and simplifies it. The vertex clustering compression technique does exactly this but may lead to some topological changes and inversion of triangles of the mesh. Also, the error in the approximations produced by schemes based on vertex clustering is usually much larger than that of the iterative edge contraction schemes and may not be acceptable.

2.1 The Core Idea

We make the observation that at any time during the simplification the set of edges collapsed by any edge collapse algorithm form a forest on the mesh graph. Each tree in this forest corresponds to a set of vertices that have been collapsed together at that time. We are working on the developing a simplification scheme that, in a first pass over the mesh, marks out a set of edges of the mesh that it then collapses after that pass. To make sure that the edges we mark are legal, we need that the marked edges should not form a cycle in the mesh graph. We ensure this by first picking a tree on the surface of the mesh and then choosing the edges to collapse from the set of edges of this tree. Any manifold mesh of genus 0 with n vertices has $3n - 6$ edges and a tree on this mesh has $n - 1$ edges. Hence, by picking a tree on the mesh we reduce the number of candidate edges by a factor of approximately 3. Now, we make passes

over this edge list and mark edges according to some heuristic. Then we collapse all these edges together in one step to get a smaller approximation of the mesh.

2.2 Initial Tree Selection

In a first pass we assign costs to each edge using the quadric error metric of [1]. A very brief description of this metric is provided for completeness. For every triangle of the mesh we can define a *fundamental quadric* Q for which $Q(v)$ is the squared distance of the point v from the plane defined by this triangle. The *fundamental quadric* Q_u for every vertex u is the weighted sum of the fundamental quadrics of its adjacent faces and $Q_u(v)$ represents the weighted sum of squared distances from v to the corresponding faces adjacent to u . When we collapse an edge (v_1, v_2) we sum up the quadrics Q_{v_1} and Q_{v_2} to get a quadric $Q_{v^*} = Q_{v_1} + Q_{v_2}$ and find the vertex v^* that minimizes Q_{v^*} . The vertex v^* is used as the location for the new vertex formed after collapsing the endpoints of the edge and the quadric Q_{v^*} is associated with this vertex v^* for further processing. The cost associated with this edge collapse is just the value $Q_{v^*}(v^*)$. The interested reader can look at [1] for more details on this error metric.

The edges of the initial tree form the set from which we will pick edges to contract later so we would like these tree edges to have as small a cost as possible. For our current experimentation we use the *Minimum Spanning Tree* of the mesh graph as our initial tree. Computing this is not a viable option when we deal with very large meshes but for the moderate sized models used for our current experiments this is not a problem. When moving to huge models we plan on replacing this with a heuristic depth first search and pick a tree that is not an *MST* but has a small cost.

Figure 1 (right) shows the *MST* computed on the mesh of a cow shown in Figure 1 (left).

2.3 Edge Selection

We have experimented with a large number of edge selection schemes to find the best edge sets to contract. I list the important ones below.

- **Even Depth Edge Contraction:** This was the first algorithm implemented. We pick a random vertex of the selected tree as the root and do a Breadth-First-Search on it labelling every vertex with its depth(distance from the root). Then we contract all the edges between vertices at depth $2d + 1$ and $2d + 2$ for all $d \geq 0$ in a single step. This gives us an approximation of the original mesh. Now the height of the tree gets reduced by half and we can apply the same procedure again to get another even smaller approximation. This can be continued until the model reaches the desired size.

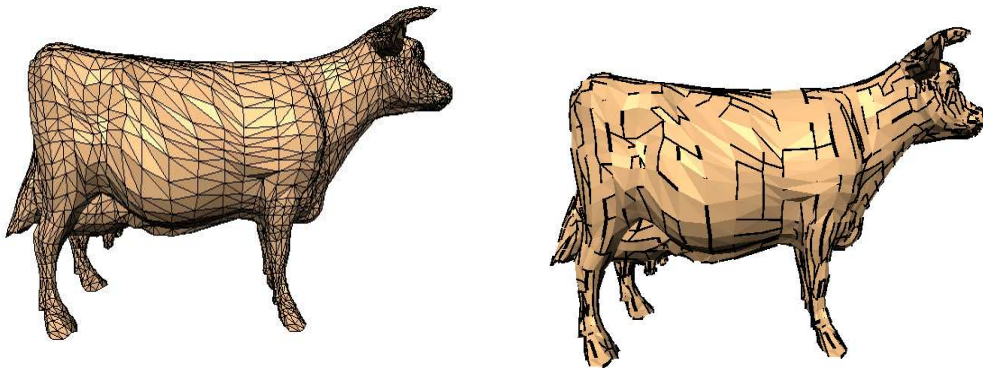


Figure 1: Left: The original mesh of the cow, Right: Spanning Tree

- Matching Selection:** Here we select a matching from the list of edges of the tree and contract those edges in a single step. This makes sure that the edges that we collapse in a single step are farther away from each other. Even though implementing this sort of a Matching based scheme for very large meshes will again be impractical this provides a good standard to measure the performance of other schemes. Various matching selection procedures were implemented. One of them just selects a maximal matching at random from among the edges of the tree. Another does a *BFS* on the tree and marks out independent edges. We experimentally determined that the maximal random matching selection algorithm was performing the best out of these matching based algorithms.

2.4 Evaluation

We used the E_{avg} metric of [4] to measure the error of the simplified model from the original model.

$$E_{avg}(M_1, M_2) = \frac{1}{w_1} \int_{v \in M_1} D_{M_2}(v) + \frac{1}{w_2} \int_{v \in M_2} D_{M_1}(v)$$

where $D_{M_1}(v)$ is the euclidean distance between point v and any point on the mesh M_2 and $D_{M_2}(v)$ is the euclidean distance between v and any point on the mesh M_1 . This metric is prohibitively expensive to compute exactly and so we compute an approximation to it by sampling the models only at the vertices. Here the weights w_i become the number of vertices in the the model M_i . For more details of this metric the reader is encouraged to look at [4].

The performance of the various edge selection schemes was compared to the iterative edge contraction scheme using quadric error to determine the least cost edge at each step. Another set of approximations was generated by using the iterative edge contraction scheme itself but restricting the set of edges used for contraction to be only the edges of the tree picked during the first phase of our algorithm. This provided a useful standard for comparison since the algorithms on the later stages also have only these edges as candidates to pick from. Also, comparing the approximations generated by this restricted iterative edge contraction scheme to those generated by the full iterative edge contraction scheme gives us an estimate of how much we lose out in the initial phase itself when we restrict ourselves to only the tree edges. Our current results indicate that working with just the tree edges is not too much of a problem by itself. We also generated approximations using the multiphase approach [3] which uses a vertex-clustering phase during which it accumulates quadric information and then follows it with an iterative edge contraction phase. These approximations provide us with a standard that we want our algorithm to perform better than.

2.5 Implementation Details

My implementation is based on the *libmesh* and *libgfx* libraries that have been developed by Prof. Garland's group at the Univ. of Illinois Urbana Champaign. *Ultra-slimfast*, an implementation of the multiphase approach was used to generate the approximations for comparison with the other schemes implemented. The library *libspace* was used for measuring the error between the different approximations and the original model.



Figure 2: The original mesh: 48K triangles

3 Results and Conclusion

The different schemes were tested on a variety of different moderately sized models. The model in Figure 2 contains $48K$ triangles. The approximations with $5K - 6K$ triangles generated by some of the schemes are shown in Figure 3.

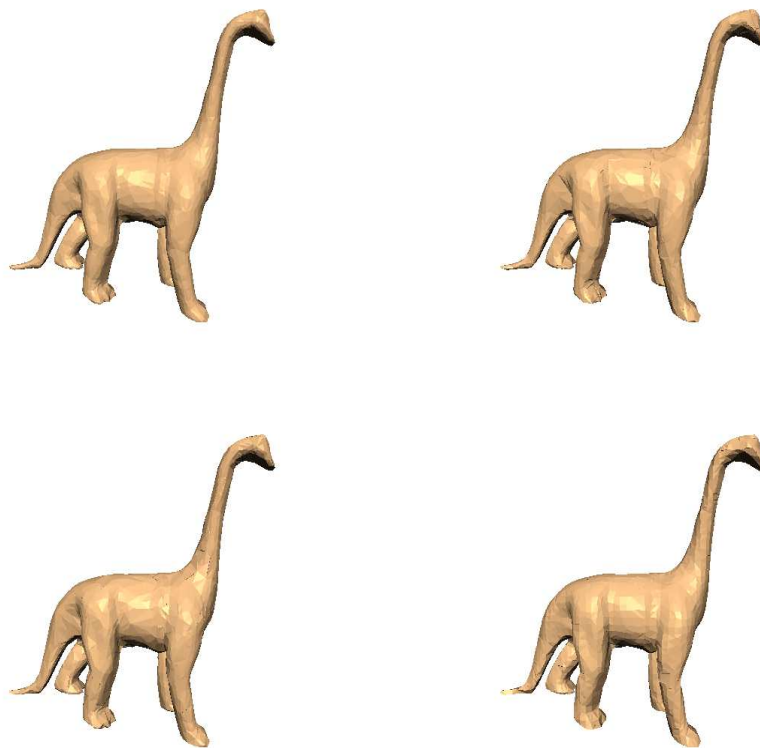


Figure 3: The approximations: Full Iterative Edge contraction (top left), Even Depth Edge Contraction (top right), Random Matching Based (bottom left), Multiphase (bottom right)

A plot of the error between the approximation and the original model against the number of edges in the approximation is shown in Figure 4 for the all the different simplification schemes mentioned above. This graph is plotted on a log scale on both axis. The error is measured as a ratio of the length of the diagonal of the axis aligned bounding box on the original model. We can note from this graph that there is a very small error difference in the curves for the full iterative version and the version that is restricted to only the tree edges. However, the even depth edge contraction and the random matching contraction schemes are not performing in comparison to the iterative version. The error for these schemes is almost comparable to the multiphase

approach. Hence, there is scope for improvement here.

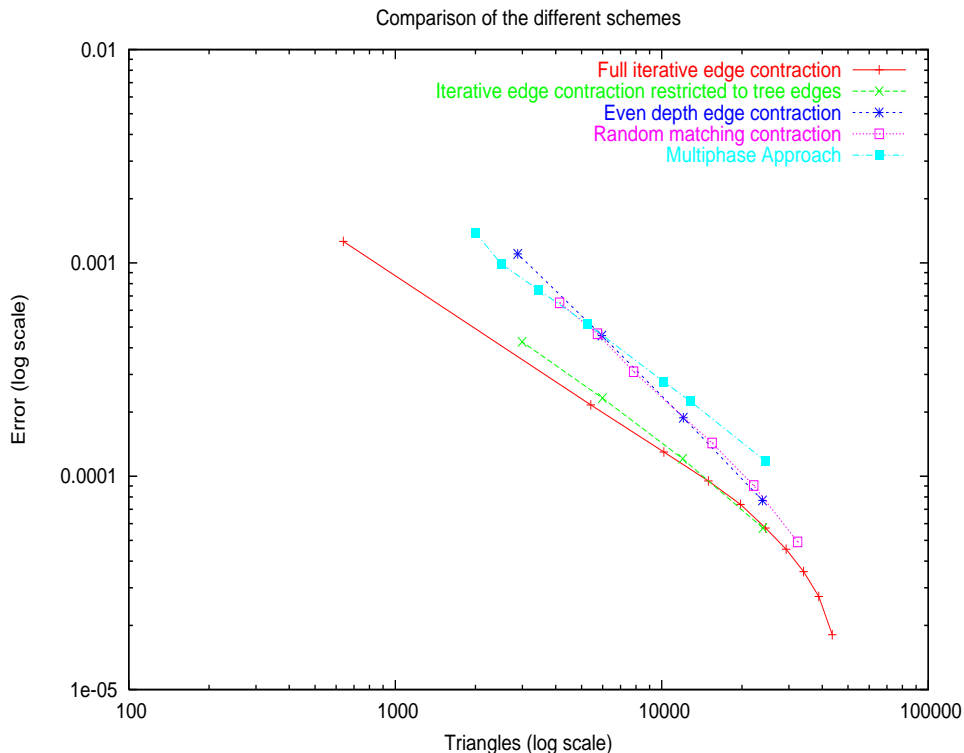


Figure 4: Error Comparison

4 Current and Planned Research

As noted in the previous section there is a large gap between error in the model produced by the iterative edge contraction scheme restricted to just the tree edges and the error in the models produced by the even-depth edge contraction or matching selection schemes. This gap exists because we do not take the cost of the edges into account after the initial tree selection process. Since, we are planning on using this technique for huge models we cannot pick only the minimum cost edge in each iteration but we can still do some sort of filtering. The idea here is that while considering the edges that we would be contracting in the next phase we would like to mark only the edges that fall below some cost threshold. This cost threshold could be the average cost of the leftover edges or some other statistic which can be computed in a single pass over the edges. This is one of the current directions on which we are working.

Also as I mentioned earlier, *MST* computation will not be a practical option for large mesh data and we would need to replace it with some other heuristic based

algorithm that would generate a tree of small cost. We would also like the tree we pick to have certain other properties like small maximum degree for its vertices (so that we do not collapse to many vertices near a high degree vertex). This problem is *NP*-complete so we would like a heuristic based that could keep the maximum degree of the vertices of the tree as small as possible. This is another direction for future work.

References

- [1] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In SIGGRAPH 97 Proc., pages 209-216
- [2] P. Lindstrom. Out-of-core simplification of large polygonal models. Proceedings of SIGGRAPH 2000, pages 259-262, July 2000.
- [3] M. Garland and E. Shaffer. A Multiphase Approach to Efficient Surface Simplification. In Proceedings of IEEE Visualization 2002, October 2002.
- [4] M. Garland. Multiresolution Modeling: Survey & future opportunities. Eurographics 99, State of the Art Report, September 1999.