Critique of Interactive Multiresolution Mesh Editing

Adam Altman
Stanford University

February 16, 2000

Abstract

This critique summarizes the ideas presented in *Interactive Multiresolution Mesh Editing*, presents this authors opinions on the material presented in and the style of the article, and poses a few questions regarding the article.

1 Paper Citation

Zorin, Denis; Schröder, Peter; Sweldens, Wim. Interactive Multiresolution Mesh Editing. Proc. Siggraph 1997, pp. 259-268.

2 Synopsis

Interactive Multiresolution Mesh Editing presents a system of describing complex geometric models using multiresolution meshes. Subdivision and smoothing algorithms are used to create a system for interactively editing the meshes. The system is designed to allow for large scale editing of meshes as well as adjustment of details. Algorithms in the system are designed to adapt themselves to the available hardware resources, allowing for reasonable editing performance on small workstations.

3 Summary

Models used for special effects, animation, and other applications often have complex geometries and varying levels of detail at different areas. Because these models are often represented as meshes of millions of triangles, interactive editing of the meshes is often difficult to do. Any editing system should also allow designers to edit large parts of the mesh smoothly as well as perform adjustments to smaller details in the model.

3.1 Earlier Mesh Editing Approaches

Other mesh editing schemas that had been tried previously include H-splines and wavelets. Both schemes have their advantages and disadvantages. Neither of these schemes lend themselves well to interactive editing of meshes.

3.2 Representing Surfaces

Surfaces may be represented in many different ways. Two of the most popular methods are polygon meshes and polynomial patches.

Polynomial patches allow designers to construct smooth models using a relatively small number of control parameters. Also, with hardware acceleration, there are fast implementations of patch algorithms available. However, polynomial parameters cannot describe all the topologies designers may want to use and, as required detail gets finer, the computation required for patches get prohibitively expensive.

Unlike patches, polygon meshes are not limited to polynomial descriptions and can be used to describe any topology at an arbitrary level of detail. However, meshes can get extremely large, making interactive editing of models difficult or impossible on current hardware.

3.3 Subdivision Surfaces

Subdivision is a scheme that falls somewhere between patches and polygon meshes in the spectrum of surface representations. This scheme defines a surface as the limit of a series of mesh refinements. Thus, subdivision forms the foundation of the hierarchical polyhedral mesh representation that allows for the large-scale manipulation that patches allow as well as the minute detail control allowed by polygonal meshes.

Four features of subdivision surfaces that are important are topological generality, multiresolution capability, simplicity, and the uniformity of their representations.

Subdivision surfaces are defined by a *graph* and a *mesh*. The graph is an abstract representation of the mesh, which is the model of the object in question. Each vertex on the graph corresponds to a point on the mesh. *Triangles* are faces of the graph or polygons in the mesh. Tangent and normal vectors are also required in order to define a local frame of reference. It is not important, though, to keep these vectors accurate through the editing process.

In a *1-ring* subdivision scheme, points at a given level of refinement only depend on 1-ring neighborhoods of points in the prior level of refinement. If the point in question lies at a vertex in the prior level or refinement, then its position only depends on positions of its immediate neighbors. If the point in question lies on an edge in the prior level, then its position depends on the positions of the neighbors of the two vertices at the endpoints of that edge. Two examples of 1-ring schemes are *Loop* and *Butterfly*. The scheme used in this system of mesh editing is the Loop scheme.

3.4 Smoothing

While subdivision allows the building of finer meshes from coarser ones, a method of building coarse meshes from fine ones is required. This operation, called *smoothing*, is a linear operation on a fine mesh that results in a mesh *one level* smoother. Least squares and global fairing approaches to this problem are far too computationally intensive to allow for interactive editing. A local smoothing technique attributed to Taubin, however, is not prohibitively expensive computationally.

3.5 Algorithms

The editing system uses two basic algorithms: *Analysis* and *synthesis*. Analysis uses subdivision and smoothing operations to calculate "detail vectors." Synthesis reconstructs a level of refinement given one coarser level of refinement. However, time and storage costs grow exponentially with the number of detail levels. Without some modifications, these basic algorithms would not allow for interactive editing for those reasons.

In order to save computation time and storage space, *adaptive* versions of the basic analysis and synthesis algorithms must be applied *locally*. Adaptive analysis allows areas without details to remain unrefined. Adaptive synthesis refines the mesh until it has reached a certain level of "local flatness."

Also, during an edit, not all of a given model changes. This allows for more optimizations by using *local* versions of these algorithms on those areas that are being edited.

Performance is improved further by using *adaptive rendering* to display the models. This only renders those triangles required to show the details of the model on the given hardware.

3.6 Results

Test meshes show how, given a limited number of triangles, the adaptive algorithms can create more detailed models by only refining those areas that need it. Performance figures were reasonable, yielding five frames per second with 113,000 triangles in a mesh on an SGI workstation.

4 Comments

4.1 Contributions

The major contribution of this paper is a new scheme for interactively editing meshes of complex models. In the process, it also introduces a technique for selectively refining areas of meshes based on the level of detail required and the hardware resources available.

4.2 Issues

The authors take great pains to ensure readers understand the details of subdivision and the subdivision techniques used in their editing scheme. However, they seem to gloss over smoothing, which may arguably be a more important aspect of their system. Instead of thoroughly explaining the process of smoothing and the technique that they used, as they did for subdivision, the authors merely point the user to another paper by a different author for details.

At first glance, it is unclear how *adaptive synthesis* is more efficient than *synthesis*. The algorithm description states that triangles are still refined, but this refinement is only temporary and some of those refined triangles are later "unrefined." Perhaps a better description of this "refinement-unrefinement" process or a better description of and comparison with the non-adaptive algorithm is in order. It should be clearly stated that, while an individual run of *adaptive synthesis* may take more time than an individual run of *synthesis*, future runs of *adaptive synthesis* will run faster because there are fewer triangles that the algorithm has to deal with.

The description of their *adaptive rendering* technique leaves much to be desired. They claim that their algorithm adapts itself not only to the level of detail required in the model, but also to the rendering performance available. Yet, they make no attempt to describe how the algorithm determines or adapts to "rendering performance." How is a balance struck between required detail and rendering performance? Are readers to believe that this system allows users to "have their cake and eat it too," so to speak? One gathers from their descriptions of their results that the algorithm itself does not "adapt" to rendering performance but that the user "adapts" the algorithm by supplying threshold parameters.

Some of the figures in the paper are too small to be of use. For example, there is a figure in the paper of three wire-frame renderings of an "armadillo." It is difficult to see the differences between the renderings and the "control points" they are referring to on images of this size.

4.3 Questions

Being able to edit meshes of complex objects interactively is a noble goal, but where did the authors get the metric of "five frames per second" being required for interactive editing? This value seems quite arbitrary and, in my opinion, still too slow for true interactive editing. What do the authors really mean by "interactive?" One can inadvertently move a control point quite "far" in one-fifth of a second with a digitizer or mouse. Or, did the authors have some other sort of input device in mind for interactive editing? If so, what?

The authors state that the basic algorithms that they used are still too computationally intensive to be used for interactive editing. They had to write adaptive and local versions of those algorithms to achieve their desired level of performance. Some of the other algorithms mentioned in the paper could not have been implemented in an adaptive manner, but others may have. Have local and adaptive versions of other systems been tried? If so, what were the results? I find it hard to believe that the authors only tried using this one set of algorithms before coming to their conclusions. What "blind alleys" did the authors follow before finding their system? Did any of these failed implementation attempts show any promise?