Critique for Progressive Forest Split Compression

Sam Liang

February 17, 2000

1 Citation for Paper

Gabriel Taubin, Andre Gueziec, William Horn, Francis Lazarus. **Progressive Forest Split Compression.** Proc. Siggraph 1998, pp. 123-132.

2 Synopsis

This paper introduces the Progressive Forest Split (PFS) representation, a new adaptive refinement scheme for storing and transmitting manifold triangular meshes in progressive and highly compressed form. The PFS scheme can smoothly interpolate between consecutive levels of detail. In addition, it achieves high compression rate — a forest split operation doubling the number n of triangles of a mesh requires a maximum of approximately $3.5n$ bits to represent the connectivity changes. The paper also shows how any surface simplification algorithm based on edge collapses can be modified to convert single resolution triangular meshes to the PFS format.

3 Summary

3.1 Problem Statement

Input A triangular mesh, defined by a set of vertices, and a set of triangles.

Output A compressed multiresolution version of the mesh in the form of a low resolution triangular mesh followed by a sequence of refinement operations.

3.2 Basic Ideas

A polygonal model is defined by the position of its vertices (geometry), and by the association between each face and its sustaining vertices (connectivity); and optional colors, normals and texture coordinates (properties). This paper concentrates on manifold polygonal models described by triangular meshes without attached properties.

The basic idea of the PFS scheme is to represent the triangular mesh as a low resolution polygonal model followed by a sequence of refinement operations. The scheme

compresses the initial low resolution model with the Topological Surgery Representation, and encodes nicely the subsequent refinement operations.

3.2.1 Topological Surgery Representation

The paper first reviews the Topological Surgery (TS) representation for a simple mesh, that is, a triangle mesh with sphere topology. The TS concept is both important for encoding the coarse mesh and for understanding the subsequent refinement operations.

The vertices of a triangular mesh are organized as a rooted spanning tree in the graph of the mesh, called the vertex tree. When a simple mesh is cut through the vertex tree edges, the connectivity of the resulting mesh is a simple polygon. The edges of the simple polygon form a boundary loop. The order of traversal of the vertex tree defines a one-to-two correspondence between the edges of the vertex tree and the edges of the boundary loop. The correspondence defines which pairs of boundary loop edges should be stitched together to reconstruct the connectivity of the original mesh.

The encoding of the representation in the compressed data stream is composed of: the encoding of the vertex tree, the compressed coordinate information, the encoding of the simple polygon in the form of a triangle tree. The vertex tree is run-length encoded. The coordinate information is placed in the form of errors in the compressed stream in the order of traversal of the vertex tree. The dual graph of the simple polygon is a triangular tree, and it is encoded similar to a vertex tree.

3.2.2 Forest Split Operation

Forest Split Operation is specified by a forest in the graph of vertices and edges of the mesh, a sequence of simple polygons (triangulated with no internal vertices), and a sequence of vertex displacements. The mesh is refined by cutting the mesh through the forest, splitting the resulting boundaries apart, filling each of the resulting tree boundary loops with one of the simple polygons, and finally displacing the new vertices.

3.2.3 Compression, Encoding and Decoding PFS meshes

The compression/encoding of the first (lowest) resolution level of detail uses the TS Representation. This block of data is followed by the compressed/encoded forest split operations in the order they are applied.

The encoding of each forest split operation is composed of, in order: 1) the encoding of the forest of edges, 2) the encoding of the sequence of simple polygons, and 3) the compression/encoding of the vertex displacements.

A simple encoding of the forest is used. It requires one bit per edge, for example a value 1 for the edges which belong to the forest and 0 for the rest.

To encode the simple polygons, the PFS scheme computes two encodings using a variable-length encoding scheme as in TS and a constant-length scheme, and selects the shorter one.

To encode the vertex displacements, their positions are first quantized to a certain number of bits per coordinate with respect to a global bounding box enclosing all the

levels of detail. The differences between these values are then Huffman encoded.

This compression scheme can achieve 3.5 bits per triangle added to the mesh for one refinement operation.

3.2.4 Conversion To PFS Format

The paper shows how any surface simplification algorithm based on edge collapses can be modified to convert single resolution triangular meshes to PFS format. The modifications require performing two simple additional topological tests on each candidate edge collapse.

The first test determines whether or not the connected components of the set of collapsed triangles are simple polygons after the edge is collapsed. The second test prevents vertices from being shared by two or more connected components.

3.2.5 Implementation and Results

Their implementation is composed of three programs: 1) a simplification program, 2) an encoding program, and 3) a decoding program.

They used the variable tolerance simplification method of Gueziec with the modifications with the two tests. The results show that the compression rate is between 3.5 to 5.5 bits per additional triangle in a refinement step, and it is between 17 to 25 bits per triangle for the entire mesh at the highest resolution level.

4 Comments

4.1 Contributions

The major contribution of this paper is the Progressive Forest Split scheme which implements a progressive scheme with high compression rate.

Compared to the Progressive Meshing scheme by Hoppe, PFS achieves higher compression rate, with fewer levels of detail. The fewer levels of detail is probably more useful than the large number of levels of details in the PM scheme.

4.2 Some Issues

4.2.1 How to Determine the Right Level of Detail?

A PFS refinement step is equivalent to a sequence of consecutive edge collapses.

They modified Gueziec's surface simplification program to generate several levels of details for a number of models. But they didn't say anything about how to determine the best number of levels of detail to generate. Also, they didn't say anything about at which point to make one level of detail.

This question may be application dependent. But they should at least present some discussion about it, and suggest some guidelines.

One possibility is that to look at the approximation error between each two consecutive tree edge cuts and refinements, and select the refinement when the approximation error suddenly jumps up.

Another possibility is select exponential number of basic refinement steps to form each level.

Any other ideas?

4.2.2 Fragile Encoding

They designed some really neat algorithms to encode and compress the data and achieved very good compression ratio.

On the other hand, the data is "squeezed" so dry, that there is almost no redundancy in it. Also, a lot of information later in the stream is highly dependent on the previous information placed earlier in the stream, so there is a very high rate of interdependencies. The risk of this is that if some part of the data, even just one bit, is corrupted or lost, the entire model may become garbage.

Of course this is an issue for most compression schemes. But it will be nice to design some schemes that has certain fault tolerance or localized error propagation characteristic, and at the same time achieves good compression ratio.

4.2.3 Some Presentation Problems

Several figures are hard to comprehend. Figure 1 should be moved to a later stage. It is not explained until section 3.2. The caption for the figure is pretty confusing at first, if you don't anything about Topological Surgery.

In figure 1(C): what are the red and black patches? No explanation.

Figure 2: the figure is rotated from D to E. Just look at the positions of the colored edges.

In figure 4 C, D: what are the dangling red tree edges?

What's a forest? The paper does not explain that it's a bunch of trees. It may be obvious to a person experienced in graph theory, but it will be nice to briefly explain it.

The Topological Surgery scheme is very hard to understand without reading the previous paper by Taubin.

5 Discussion Questions

How did they come up with the algorithm? I want to see a section in all papers to talk about how they got the inspiration, how God dropped the idea onto them from the heaven.

How do you come up with a fancy name for your algorithm, like Topological Surgery?