Streaming Simplification of Tetrahedral Meshes

Huy T. Vo[1], Steven P. Callahan[1], Peter Lindstrom[2], Valerio Pascucci[2], and Cláudio T. Silva[1]

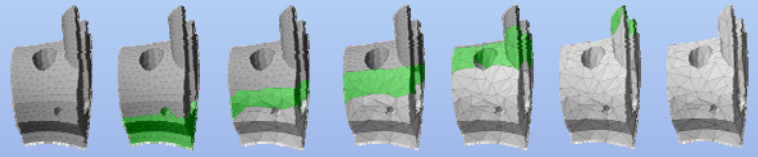
[1] Scientific Computing & Imaging Institute, The University of Utah

[2] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory

Unstructured tetrahedral meshes are commonly used in scientific computing to represent volume dataset. However, it is hard to visualize them interactively due to their size and complexity . By reducing the size of the data, we can accomplish real-time visualization necessary for scientific analysis. Our algorithm first arranges the data on disk in a streaming, I/O-efficient format. Then, a quadric-based simplification is sequentially performed on small portions of the mesh in-core. Our output is a coherent streaming mesh, which facilitates future processing. Our technique is fast, produces high quality approximations, and operates out-of-core.

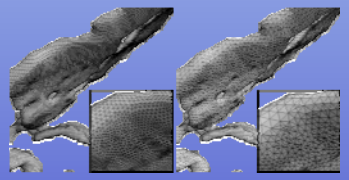
Introduction:

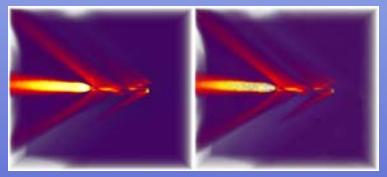
- Our method does not require the entire mesh to remain in main memory, but only a small portion of the mesh instead
- Represent the mesh in a coherent sequential access format
- Output mesh is also in a coherent format
- Using a new solver to improve stability and performance



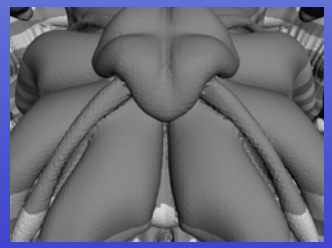
Results:

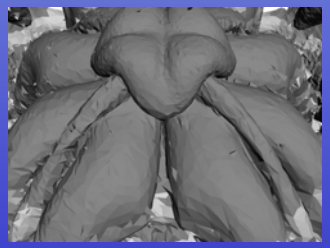
All of our results show that our method can produce approximations with negligible error for both scalar field and surface geometry. In fact, we were able to simplify meshes containing up to 1 million tetrahedra to 1 percent with only 1.5% root-mean-squared error.





Surface and scalar field quality of the resulted meshes





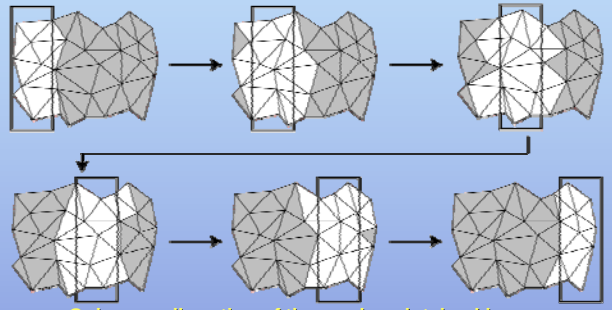
Iso-surfaces extracted from the PPM dataset consisting of more than a billion tetrahedra

Methods:

INPUT: *data set* and *simplification ratio* OUTPUT: *low resolution mesh*

3-stage algorithm with a fixed size buffer:

- Read and reduce the buffer to 50% until we reach the desired ratio
- Read, reduce the buffer to meet the *ratio* and only keep half the buffer in-core until we finish reading
- Simplify the current buffer to the desired *ratio* and write everything out

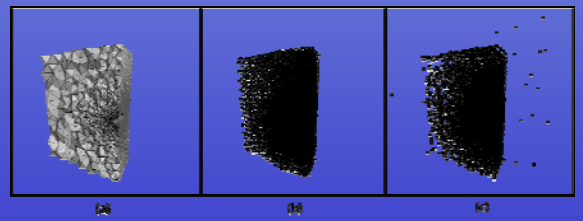


Only a small portion of the mesh maintained in-core

We use a randomized multiple choice quadric based for edge-collapse simplification

Discussion:

- Enhance stability and performances of *CG* vs. *Cholesky* method
- Advantage of the new quadric function representation (A, p, e)
- Improve performance and memory footprint with *Corner Table* data structure
- Store meshes in binary format for fast I/O
- Simple and fixed-size data structure for each vertex/tetrahedra



Stability of *CG* vs. *Cholesky*

```
VERTEX
    matrix A          10 floats
    vector p          4 floats
    float e
    int idx           global index
    int corner        vertex-to-corner index
    uchar flags       deleted, written
TETRAHEDRON
    int vidx[4]       vertex indices
    bool deleted
    int order         for ordering tetrahedra
    int L[4]          corner links
```

Main data structure

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48

UCRL-POST-214382