Constructing Tetrahedral Meshes No Matter How Ugly The CAD

Matt Staten* David Noble* Riley Wilson*

Abstract
We present the progress on the development of a tetrahedral mesh generator that will automatically generate fully conformal tetrahedral meshes on industrial CAD assemblies, regardless of how dirty and incorrect the CAD description is, without any user cleaning up or tweaking of the CAD before mesh generation.

1 Introduction

Traditional bottoms-up tetrahedral meshing, which puts nodes on CAD vertices, followed by edges on CAD curves, then triangles on CAD surfaces before finally filling the interior with tetrahedral elements, requires the CAD to be *perfect*. Industrial CAD models routinely have dozens to hundreds of errors including overlaps, incorrectly-sized gaps in assembly components, CAD curves out-of-tolerance with neighboring CAD surfaces, missing surfaces resulting in a non-water tight boundary, excessively detailed CAD features, high order spline surfaces that stop imprinting of neighboring assembly components when simple analytics will suffice, etc. These errors must be removed from the CAD before traditional tetrahedral meshing methods can be employed.

While significantly less arduous than the CAD cleanup required for hexahedral meshing, removing all of these CAD errors to achieve the required level of CAD *perfection* for traditional tetrahedral meshing can still take many hours to days of tedious manual user interactive editing and requires significant skill and attention to detail. However, design phase iterations require rapid turn around in order to thoroughly consider the entire design space. Requiring a manual user-intensive CAD cleaning step in the process is, at best, a major impediment and often a show stopper.

We are developing a geometry tolerant tetrahedral mesh generator, which we have named *Morph*, which automatically generates tetrahedral meshes without the need to remove the CAD errors. We employ *mesh based defeaturing* rather than CAD repair and defeaturing. We have chosen an overlay grid approach using node snapping similar to CISAMR [1] and Isosurface Stuffing [2], using a BCC lattice [2], and element cutting similar to the Conformal Decomposition Element Method [3].

We have adapted these approaches to also capture sharp CAD features, which was previously done with Improved Isosurface Stuffing [4]. However, we employ both snapping and cutting to capture the sharp features which significantly increases the likelihood the features will be captured with quality. TetWild [5] also captures sharp features, doing so by first capturing *all* CAD features, and then subsequently collapsing out those smaller than a prescribed tolerance. In contrast Morph introduces the ability to selectively capture large features, while at the same time automatically ignoring and never cutting in any feature smaller than a tolerance. This avoids the introduction of excessively small and often poorly shaped elements that must later be collapsed out.

Further, Morph generates meshes on CAD assemblies, using a single tet overlay for the entire assembly, partitioning the resulting tetrahedra into each assembly component. The same mechanism which ignores small CAD features is used again to automatically collapse out small gaps introduced by sloppy placement of components in the CAD assembly.

2 Morph Algorithm Overview

Our approach follows the following basic steps:

1. Create an unstructured tetrahedral overlay larger than the bounding box of the object.

2. Refine the overlay to be roughly the desired element size in each region of the mesh.

3. Cut and Snap the overlay to the CAD:

*Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energys National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government. The publisher acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this written work or allow others to do so, for U.S. Government purposes. The DOE will provide public access to results of federally sponsored research in accordance with the DOE Public Access Plan.

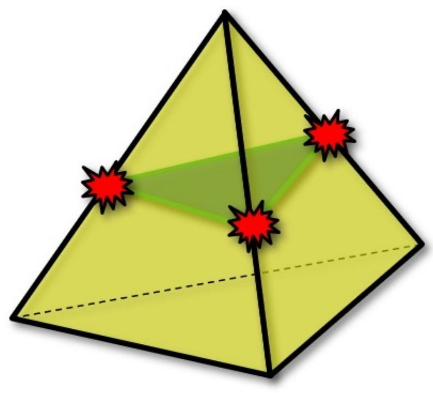

Figure 1: CAD-Surface to Overlay-Edge Intersection

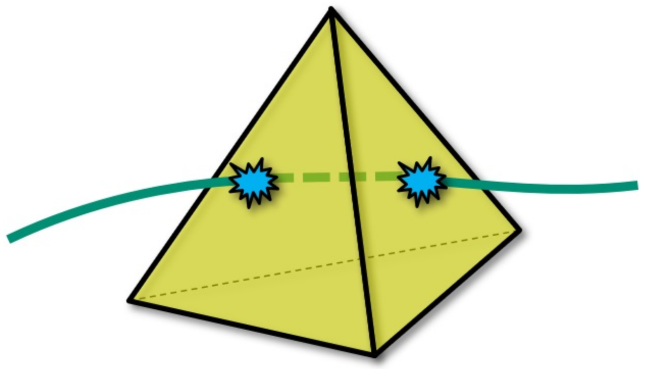

Figure 2: CAD-Curve to Overlay-Tri Intersection

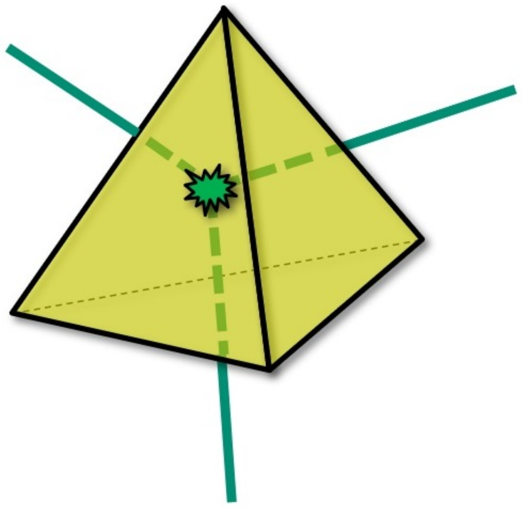

Figure 3: CAD-Vertex to Overlay-Tet Intersection

(a) Compute the *intersection points* where the CAD surfaces, curves and vertices intersect the overlay edges (fig. 1), triangles (fig. 2), and tetrahedrals (fig. 3).

(b) Assign the geometry from any intersection point within tolerance of an existing geometry-assigned node, and then throw away that intersection point.

(c) Cut new nodes and/or snap existing overlay nodes onto the remaining intersection points.

(d) Assign the geometry from each intersection point to the node cut or snapped to it.

(e) Repeat until converged.

4. Assign the tetrahedrals to the various volumes, or to the *air* around the object.

5. Perform tetrahedral quality improvement operations to improve element quality.

6. Delete the outside tetrahedrals. Optionally, keep the outside tetrahedrals and delete the interior tetrahedrals.

This approach captures the geometry to the length scale of the overlay grid and no further, automatically washing over small features, collapsing gaps, etc. smaller than the tolerance used in step 3b. In contrast, traditional bottoms-up approaches will often drill down onto small CAD features with excessively small elements, driving timesteps down for explicit analyses.

This approach also naturally and automatically removes any mesh overlap from sloppy placement of assembly components. Since there are no overlapping elements in the overlay, by definition, the resulting mesh cannot be overlapping. Overlaps smaller than the tolerance will be collapsed out. Larger overlaps will be cut in with elements that could be assigned to either of the overlapping components, which are then assigned to the one with higher priority.

3 Discussion

We have implemented this algorithm for distrubuted memory MPI HPC machines, using both the ACIS solid modeler and a Sandia internal solid modeler, as well as for STL facet files. We have integrated this tetrahedral mesh generator into both rapid turn-around design tools and design optimization loops. In addition to meshing the solid objects themselves, this approach can also be used to mesh the exterior *air* surrounding the objects. We have found success using the resulting meshes for structural dynamics, thermal, aerodynamics, and electromagentic applications. fig. 4

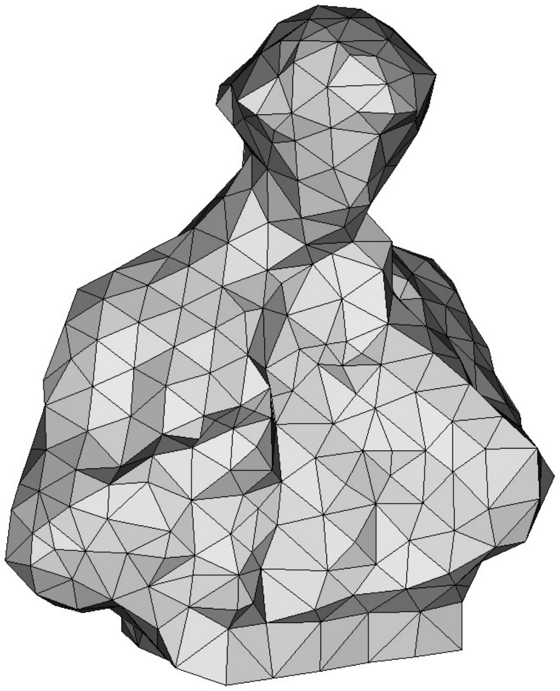

Figure 4: Edgar Allan Poe size 10.0, 2,882 tetrahedral elements, 14 MPI procs, 14 seconds generation time, min scaled Jacobian: 0.226

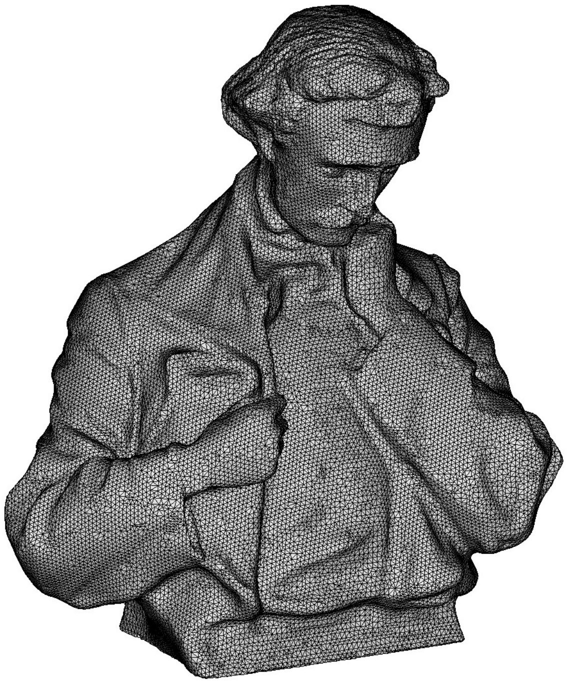

Figure 6: Edgar Allan Poe size 1.0, 2.78M tetrahedral elements, 14 MPI procs, 74 seconds generation time, min scaled Jacobian: 0.129

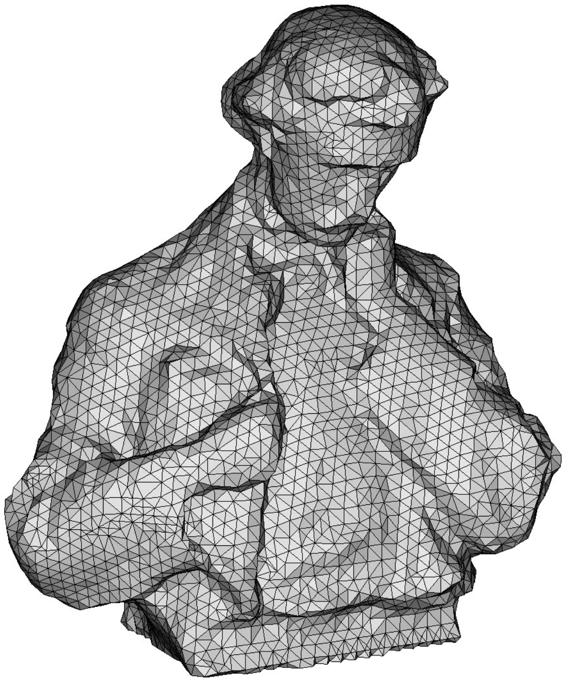

Figure 5: Edgar Allan Poe size 3.0, 103,170 tetrahedral elements, 14 MPI procs, 26 seconds generation time, min scaled Jacobian: 0.113

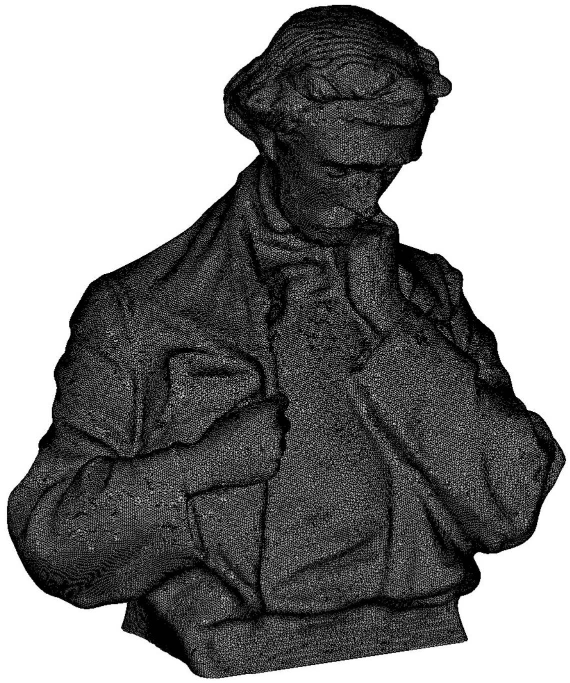

Figure 7: Edgar Allan Poe size 0.5, 22.2M tetrahedral elements, 14 MPI procs, 392 seconds generation time, min scaled Jacobian: 0.135

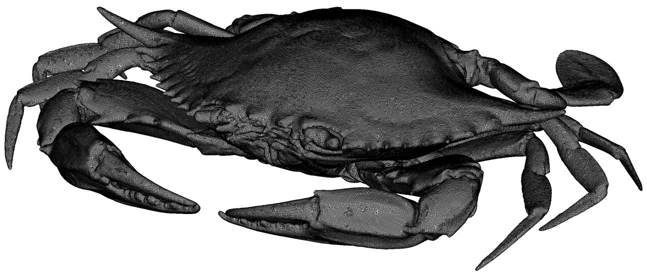

Figure 8: The IMR crab, 7.6M tetrahedral elements, 14 MPI procs, 531 seconds generation time, min scaled Jacobian: 0.126

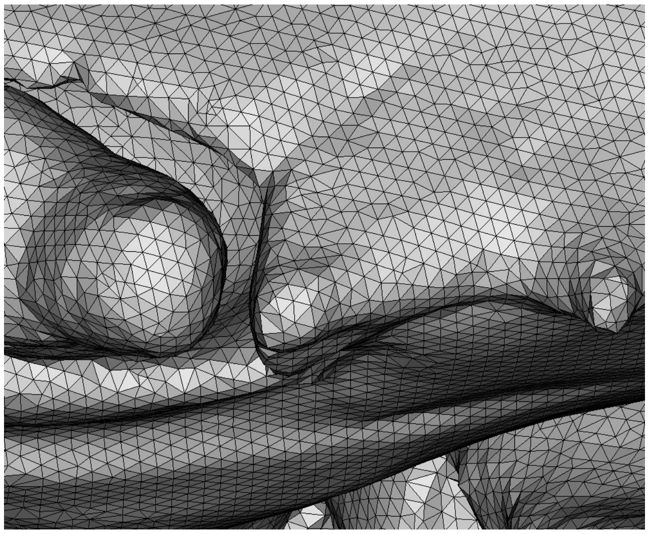

Figure 9: Close-up of the IMR crab

to fig. 9 illustrate the resulting meshes on the IMR test models.

In this presentation, we will discuss the challenges, advantages and disadvantages of this approach as well as variations of the algorithm for different applications.

References

[1] A. Nagarajan and S. Soghrati, *Conforming to interface structured adaptive mesh refinement: 3D algorithm and implementation*, Computational Mechanics, 62 (2018), pp. 1213-1238, https://doi.org/10.1007/s00466-018-1560-2.

[2] F. Labelle and J. R. Shewchuk, *Isosurface stuffing: fast tetrahedral meshes with good dihedral angles*, Proc. ACM Siggraph (2007).

[3] S. A. Roberts, H. Mendoza, V. E. Brunini and D. R. Noble, *A verified conformal decomposition finite element method for implicit, many-material geometries*, Journal of Computational Physics, 375 (2018), pp. 352-367, https://doi.org/10.1016/j.jcp.2018.08.022.

[4] C. Doran, A. Change and R. Bridson, *Isosurface stuffing improved: acute lattices and feature matching*, Proc. ACM Siggraph (2013).

[5] Y. Hu, T. Schneider, B. Wang, D. Zorin and D. Panozzo, *Fast tetrahedral meshing in the wild*, ACM Trans. Graph., 39:4, Article 117 (2020), https://doi.org/10.1145/3386569.3392385.