Massively Parallel Tet Meshing With Size-Dependent Feature Capture on CAD Models

M. L. Staten, D. R. Noble, C. R. Wilson, C. L. McBride, M. K. Bhardwaj

Abstract A snapping overlay octree grid approach to tetrahedral meshing is described. The method meets the requirements of massively parallel (distributed memory), a minimum resulting edge length, CAD models (potentially dirty) as input, and capture CAD features that are large compared to the element size but automatically ignore smaller CAD features. The goal is to simplify the meshing process by not requiring the user to make any *cleanup* changes to the CAD model to repair or remove small features. The status of the capability is described including examples.

1 Introduction

Unlike other other disciplines, explicit solid mechanics has traditionally relied on hexahedral elements for accuracy in bending and plasticity. Unfortunately, hex mesh construction on complex CAD assembly models can take months of manual tedium [1]. However, the recent development of the composite tetrahedral element [2] shows promise that a tet element may finally yield solution accuracy comparable to hex elements for solid mechanics.

Mesh developers now focus on adapting tet generators to solid mechanics requirements, including parallel scaling to thousands of processors, automatically capturing CAD features large compared to the element size, while ignoring features small compared to the element size, and honoring a minimum edge length to bound the time step for explicit analyses.

Tet meshing has been researched for decades. Bottoms up tetrahedral meshing puts nodes on CAD vertices, then edges along CAD curves, then triangles on CAD surfaces, then finally adding interior tet elements. However, this approach requires a perfect CAD model to avoid tiny edges and angles caused by tiny CAD features. Shepherd [3] introduced an overlay grid *cutting* algorithm, which requires complex cleaning after cutting to avoid sliver elements violating minimum element sizes. Nagarajan and Soghrati [4] presents a 3D overlay grid *snapping* approach for level set geometry, which could be applied to CAD. However, CISAMR applies snapping constraints that result in quality guarantees, but limit the ability to capture some simple CAD features. Labelle and Shewchuk [5] present Isosurface Stuffing, a *snapping* and *cutting* hybrid approach with guaranteed quality, but only for single isosurface

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525

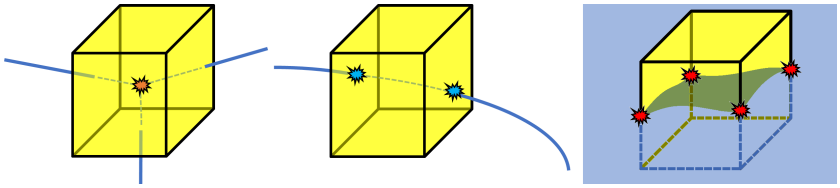

Fig. 1 The three types of intersection points. LEFT: vertex-cell intersection point where a vertex is located inside an overlay grid cell; CENTER: curve-face intersection points where curves intersect overlay grid faces. Right, surface-edge intersections where surfaces intersect overlay grid edges.

geometries. Doran [6] added CAD feature capture to Isosurface Stuffing, but as a post processing step that would be difficult to parallelize and requires smoothing and swapping to improve quality after CAD feature capture.

For this work, we have chosen a full-*snap* approach. We operate directly on CAD, rather than converting the CAD to a level set or volume fraction representation. Our goal is to eliminate any and all constraints which require user modification of the CAD before meshing, such as removing small features or making it water-tight.

First, we construct a constant-size cartesian overlay grid from the CAD bounding box, with one layer of ghosted cells. Future extensions will support refined octree grids. Second, we compute a grid to CAD intersection resulting in a set of xyz intersection points. Third, selected overlay grid nodes are *snapped* to prioritized intersection points, morphing the grid to the CAD boundary and features. Finally, the morphed overlay grid is converted into tet elements using a variation of the Body-Centered Cubic (BCC) lattice [5].

2 Intersecting CAD with Overlay Grid

Computing the intersection of an overlay grid and CAD results in a set of 3-tuples each containing an xyz intersection point, along with both the grid and CAD entity causing the intersection. Three types of intersections are computed, as illustrated in Figure 1. Efficient calculation of intersection points leverages the axis alignment of the overlay grid.

3 Snapping overlay grid nodes to intersection points

Overlay grid nodes are snapped to prioritized intersection points, *morphing* the grid to the CAD. Cells intersecting the CAD boundary usually contain many intersection points, so not all intersection points will be chosen for snapping. We prioritize which overlay grid nodes get snapped, and to which intersection points.

Snapping is done using a graph in order of increasing dimension of the geometry entity in the intesection point 3-tuple. Snapping to vertex intersection points is performed first, followed by snapping to curve intersection points, followed by snapping to surface intersection points.

Figure 2 illustrates a green geometry with just one of the many cells in the overlay grid, along with 2 vertex-cell, 2 curve-face, and 4 surface-edge intersection points.

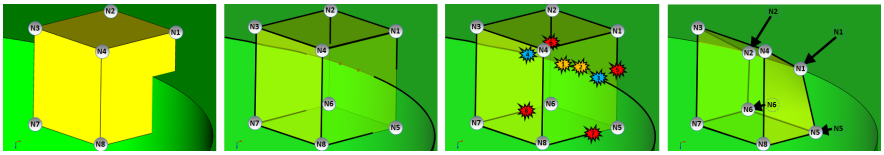

Fig. 2 Snapping example model. FAR LEFT: green geometry and a single overlay grid cell. CENTER LEFT: transparent model to see two vertices inside the cell. CENTER RIGHT: the eight resulting intersection points. FAR RIGHT: the cell morphed to the CAD.

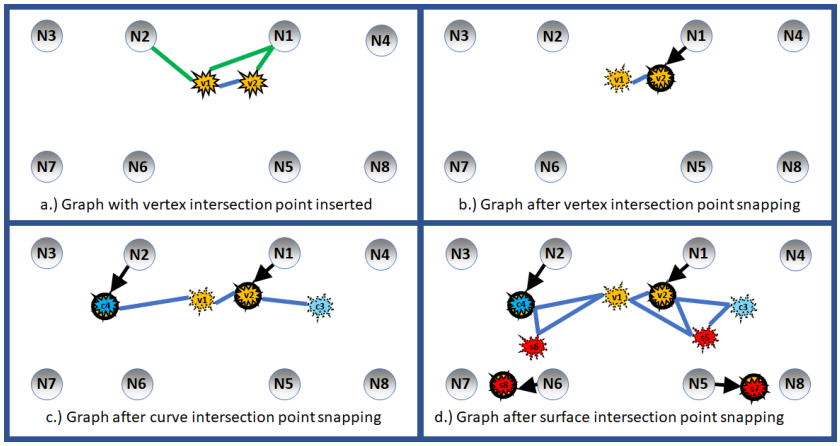

Fig. 3 Graph for snapping example.

Figure 3 illustrates the evolution of the snapping graph, considering just the intersection points from Figure 2. Figure 3a shows the graph after the vertex intersection points are inserted. Graph edges are created between each intersection point and the closest overlay grid node, or grid nodes in the case of ties. Graph edges are also added between intersection points which are within *h/2* of each other, where *h* is the side length of the local overlay grid cell.

Figure 3b shows the graph after vertex intersection point snapping illustrating that priority is given to the shortest intersection point snaps. Because N1 was snapped to v2, and because v2 is connected to v1 in the graph (i.e. within *h/2*), v1 is disconnected from all grid nodes as it is no longer a candidate for snapping. This guarantees that no two grid nodes will be snapped within *h/2* of each other. This also provides automatic washing out of small geometric features. When a grid node has more than one snapping option, the closest intersection point of the lowest geometry dimension is chosen, and the rest are ignored.

The process repeats for curve intersection points (Figure 3c) and then surface intersection points (Figure 3d). Figure 2 illustrates the final morphed grid cell. All grid nodes and intersection points across the overlay grid are included in the graph, resulting in all cells intersecting the CAD boundary being morphed to the CAD.

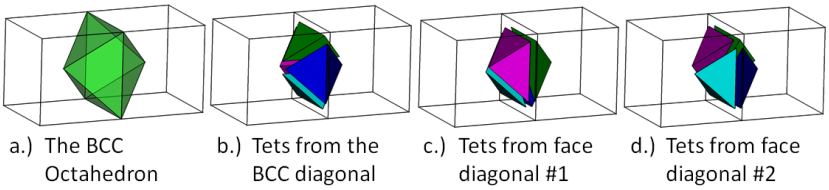

Fig. 4 An octahedron between two adjacent overlay grid cells, and the 3 diagonal choices..

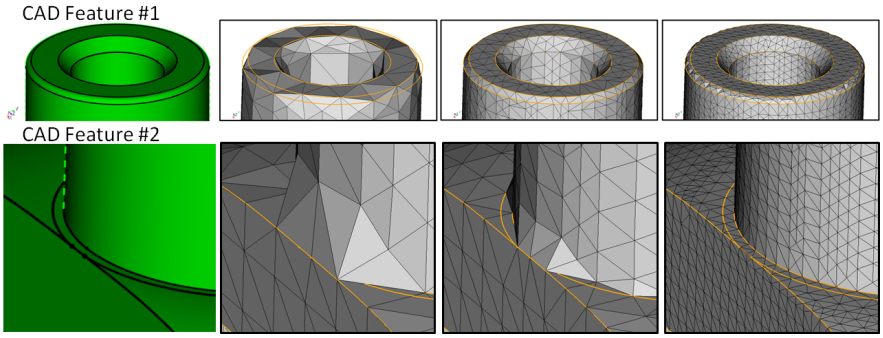

Fig. 5 Two CAD features on knuckle model meshed at 3 mesh sizes.

4 Converting Morphed Overlay Grid To Tets

Finally, the morphed overlay grid is converted into tet elements using a variation of the BCC lattice [5]. For cells completely inside the CAD, with some nodes potentially snapped to the CAD boundary, the BCC center node is placed at the centroid of the morphed cell's nodes. Cells completely outside the CAD do not need a BCC node. Snapping results in some grid cells straddling CAD curves or surfaces. In this case, the BCC node is located at the closet point projection to the straddled CAD surface or curve of the centroid of the morphed cell's nodes.

For an almost embarrassingly parallel tet creation we build tets for each face in the overlay grid. Consider an octahedron built using the 4 nodes from the face and the 2 adjacent BCC nodes (Figure 4a). The standard BCC lattice connects BCC nodes in adjacent cells with an edge and forms four tets surrounding the edge (Figure 4b). However, there are two other diagonals that could be used instead (Figures 4c and d). We choose which diagonal to use based on several criteria. First, some of the octahedron nodes may not be available, for example, if a BCC node was not created. Second, one of the face diagonals may be required if the face straddles the CAD boundary. Third, the diagonal choice is based on the best quality tets when considering the morphed octahedron's node locations.

5 Results and Conclusion

Figure 5 shows two CAD features meshed at three grid sizes. When the overlay grid size is large, CAD features are washed away, being only partially captured because there are not enough overlay grid nodes to snap to all the geometry in the feature. As

Table 1 Quality for knuckle example model.

	#elements	Scaled Jacobian	Aspect Ratio
Coarse Size	7,051	0.152	5.88
Medium Size	63,576	0.174	6.70
Fine Mesh	277,538	-0.128	n/a

the grid size is refined, vertices and curves in CAD features are captured precisely. However, even at small mesh sizes, tangencies between CAD curves are washed out to avoid tiny element angles, as illustrated in the finest mesh on CAD Feature 2.

Table 1 shows element counts and quality on the knuckle model. On two of the meshes, the quality is good. However, on the 3rd mesh, there are 12 poorly shaped elements where a curve was captured in a concavity. This corraborates the statement from Labelle [5], who states that quality guarantees are lost if sharp geometry features are to be captured. We believe that in practice, the proposed morphing algorithm will provide reasonable quality in most cases, and localized smoothing or a few topological swaps can repair remaining bad elements. Doran [6] successfully performs swapping and smoothing after their feature capture.

The presented algorithm automatically captures CAD vertices, curves and surfaces larger than *h*root(3)*, (i.e. the diagonal length of an overlay grid cell) and automatically defeatures CAD features smaller than *h/2*. CAD features in-between are sometimes captured, and sometimes defeatured depending on their orientation with the overlay grid.

All operations in the presented algorithm are local, making this easy to parallelize. We plan to extend the proposed algorithm to add localized post-process cleanup operations to improve quality, handle multi-volume CAD assemblies and provide adaptively sized meshes by using refined octree overlay grids.

References

1. J. Shepherd, C. R. Johnson. ”Hexahedral Mesh Generation Constraints.” Engineering With Computers, 24(3) 195-213, 2008.
2. J. T. Ostien, J. W. Foulk, A. Mota, M. G. Veilleux. ”A 10-node composite tetrahedral finite element for solid mechanics” Int. J. Numer. Meth. Engng, 107:1145-1170, 2016.
3. M. Shephard, M. Georges. ”Automatic Three-Dimensional Mesh Generation By Finite Octree Technique.” International Journal For Numerical Methods In Engineering, 32:709-749, 1991.
4. A. Nagarajan, S. Soghrati. ”Conforming to interface structured adaptive mesh refinement: 3D algorithm and implementation.” Computational Mechanics, 2018, https://doi.org/10.1007/s00466-018-1560-2.
5. F. Labelle, J. R. Shewchuk. ”Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles.” ACM Transactions on Graphics, 26(3):57.1-57.10, July 2007. Special issue on Proceedings of SIGGRAPH 2007.
6. C. Doran. ”Isosurface Stuffing Improved: Acute Lattices and Feature Matching.” Master’s Thesis, University of British Columbia, Oct 2013, https://open.library.ubc.ca/cIRcle/collections/ubctheses/24/items/1.0052176.