Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 163 (2016) 251 – 261

Procedia Engineering

www.elsevier.com/locate/procedia

25th International Meshing Roundtable (IMR25)

All-Hex Meshing of Multiple-Region Domains without Cleanup

Muhammad A. Awad[a], Ahmad A. Rushdi[b,c], Misarah A. Abbas[d], Scott A. Mitchell[c], Ahmed H. Mahmoud[a], Chandrajit L. Bajaj[b], Mohamed S. Ebeida[c,*]

[a] *University of California, Davis, CA 95616, U.S.A.*
[b] *Institute for Computational Engineering and Sciences, University of Texas, Austin TX 78712, U.S.A*
[c] *Center for Computing Research, Sandia National Laboratories, Albuquerque NM 87185, U.S.A.*
[d] *Alexandria University, Alexandria, Egypt*

Abstract

In this paper, we present a new algorithm for all-hex meshing of domains with multiple regions without post-processing cleanup. Our method starts with a strongly balanced octree. In contrast to snapping the grid points onto the geometric boundaries, we move points a slight distance away from the common boundaries. Then we intersect the moved grid with the geometry. This allows us to avoid creating any flat angles, and we are able to handle two-sided regions and more complex topologies than prior methods. The algorithm is robust and cleanup-free; without the use of any pillowing, swapping, or smoothing. Thus, our simple algorithm is also more predictable than prior art.

Peer-review under responsibility of the organizing committee of IMR 25
Keywords: All-hexahedral Meshing; Mesh Generation; Guaranteed Quality

1. Introduction

We are given a 3d domain $\mathcal{D}$ of multiple regions, defined by a standard b-rep boundary representation. An all-hex mesh $\mathcal{M}$ has only hexahedral elements, and accurately represests the domain $\mathcal{D}$. The quality of the mesh plays a significant role in the accuracy and stability of numerical simulations or solution of PDEs, e.g., Finite Element Analysis (FEA) [1–3] and Computer-Aided Design (CAD) [4,5].

In 2d, unstructured all-quad meshing algorithms are usually categorized into two main categories: *indirect* and *direct*. A classical indirect approach starts with a triangular mesh, and then transforms the triangular elements into quadrilateral elements, via optimization [6,7], refinement and coarsening [8], or simplification [9]. A class of indirect methods start with a triangular mesh and applies the mid-point subdivision rule [10,11] to split a triangle into three quad elements. This can be achieved using local subdivision operations. For example, [12] implements a recursive subdivision algorithm based on a regular tiling composed of only diamonds and kites, but does not handle domain boundaries. Q-Morph [13] is a popular indirect approach that follows a sequence of systematic triangle transforma-

* Corresponding author. Tel.: +1-505-844-0456
E-mail address: msebeid@sandia.gov

Peer-review under responsibility of the organizing committee of IMR 25
doi:10.1016/j.proeng.2016.11.055

tions to create an all-quadrilateral mesh. However, Q-Morph requires topological cleanup and smoothing to guarantee the quality of the final all-quad mesh. Q-Tran [14] is another indirect algorithm that produces quadrilaterals with provably-good quality without a smoothing post-processing step, and manages to handle domain boundaries. Nevertheless, the class of indirect methods typically suffers from a large number of irregular nodes that are connected to more (or less) than four mesh elements, which is typically undesired in several numerical simulations. Direct approaches, on the other hand, construct quadrilaterals directly. The advancing front algorithms (e.g., the paving algorithm [15]) successfully generate all quad meshes with high quality, by placing mesh points on the boundaries of the input domain and form quad elements by recursively projecting edges on the front towards the interior of the domain until the whole domain is covered with quads [16,17]. However, they suffer from stability problems that require heuristic cleanup operations. Grid based methods construct a uniform Cartesian or quadtree background grid and aim at modifying that grid to conform to the domain boundaries [18,19]. These methods are easy to implement and can provide quality guarantees and angle bounds [20–22]. However, they often result in inverted elements.

In 3d, the all-hex meshing problem has not been completely solved yet. Many diverse approaches have been tried, including constructing meshes from volumetric data [23], from surface quad meshes [24], using the medial axis transform [25], using midpoint subdivision [26], singularity-restricted frame fields [27], and volumetric PolyCube deformation [28]. However, automating these types of methods [29–31] faces several challenges, especially when proper meshing constraints are taken into consideration [32]. Many methods can only achieve hex-dominant meshes [33,34] and some require parallel implementations [35]. Different applications require different mesh properties, which makes the choice of the proper meshing approach more difficult [36–38].

1.1. Grid-based Hex Meshing

Our approach belongs to the grid-based or octree family of methods [19,39–41]. The two key challenges for octrees are to capture the domain boundary with high-quality hexes [37], and to generate all-hex elements in the presence of hanging nodes arising from size transitions. In addition, these methods tend to produce a large number of hexes. To capture the boundary, a common approach is to introduce a boundary layer of hexes. *Pillowing* is one method. For each hex, one face is on the boundary and the opposite face mates with a face of the axis-aligned octree. High quality hexes are challenging with this abrupt orientation change, and smoothing is typically required. Interior to the domain, a variety of transition templates are used to resolve hanging nodes. Early work [19] had fairly restrictive templates, and these have been expanded to allow more rapid and flexible size transitions [40,41]. Hexotic [39] adds ideas from midpoint subdivision to its octree to reduce the number of hexes generated. Its dual transition templates ensure that every corner and hanging node of the octree has six edges, each of which has four faces, but the polyhedral cells are not necessarily hexes. This can be immediately dualized to form hexes, which is equivalent to performing midpoint subdivision on the hexes, then combining the eight hexes surrounding each original node.

1.1.1. Snapping

Hexotic and dual contouring, among others, *snap* (project) nodes to the geometry, then fix hex quality by boundary layers and pillowing. The fundamental shortcoming of snapping is that there is no known way to handle geometry whose local topology is more rich than the topology of the octree. That is, an octree edge has at most four faces, and what does one do if the geometry has an edge with five faces? Also nodes are limited to edge-degree six.

Snapping and our approach are fundamentally different. We do not attempt to match the grid topology to the geometry topology, but instead intersect octree cells with the geometry. This allows us to handle domains with arbitrary topology in principle.

1.2. Contribution

In this paper, we extend the recently developed cleanup-free all-quad meshing approach in [42] (extended for sharp corners in [43]), introducing a new direct grid-based algorithm that produces an all-hex mesh without post-processing cleanup. Conceptually, we follow similar steps to those in [42,43]. Starting with a Cartesian grid or an octree domain decomposition, we repel the grid points *away* from the input geometry, intersect each cube with the geometry faces, use midpoint subdivision to split the intersected elements into hexes, and finally apply 2-refinement templates to