HybridOctree_Hex: Hybrid Octree-Based Adaptive All-Hexahedral Mesh Generation with Jacobian Control

Hua Tong, Eni Halilaj, Yongjie Jessica Zhang
Computational Biomodeling Laboratory, Department of Mechanical Engineering, Carnegie Mellon University

Introduction

We introduce "HybridOctree_Hex," a new software for adaptive all-hexahedral mesh generation. It uses a hybrid octree approach with Jacobian control for quality enhancement. Key surface features are identified from input boundaries to initialize the octree. Balanced and paired, pre-defined templates enable direct generation of dual meshes. Elements outside the boundary are removed to form a core mesh. The gap between the core mesh and the input domain is then filled to create the final mesh. Smart Laplacian smoothing and optimization-based quality improvement techniques ensure a minimum scaled Jacobian above 0.5. Our method is robust and efficient, proven on dozens of complex 3D models without manual intervention. The mesh results and source code are available at: github.com/CMU-CBML/HybridOctree_Hex.

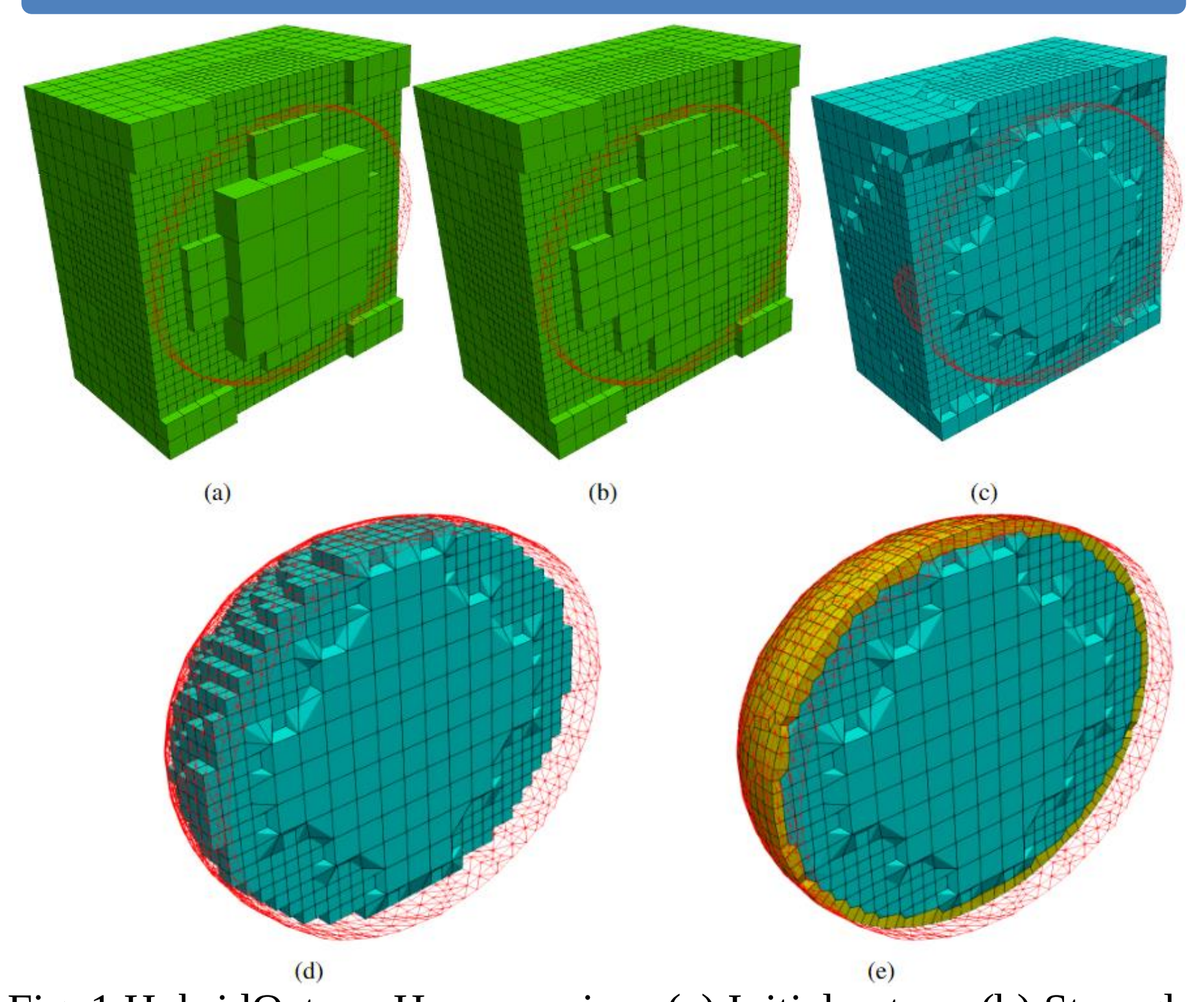


Fig. 1 HybridOctree_Hex overview. (a) Initial octree; (b) Strongly balanced octree; (c) Construct all-hex dual mesh using pre-defined templates; (d) Core mesh; (e) Meshing the buffer zone (yellow) with geometric fitting and Jacobian control.

Initializing the Octree and Building A Strongly Balanced Octree Structure

Given a smooth boundary, we adopt a Gaussian curvature function and a thickness function to capture and refine high curvatures and narrow regions on the surface. The octree is refined to a deeper level as the curvature/ thickness becomes higher/ smaller.

The following two rules are applied to convert the octree to a strongly balanced one.

Balancing rule: ensures that the level difference between two neighboring octants is at most one.

Pairing rule: if an octant is subdivided to comply to the balancing rule, its siblings are subdivided along with it.

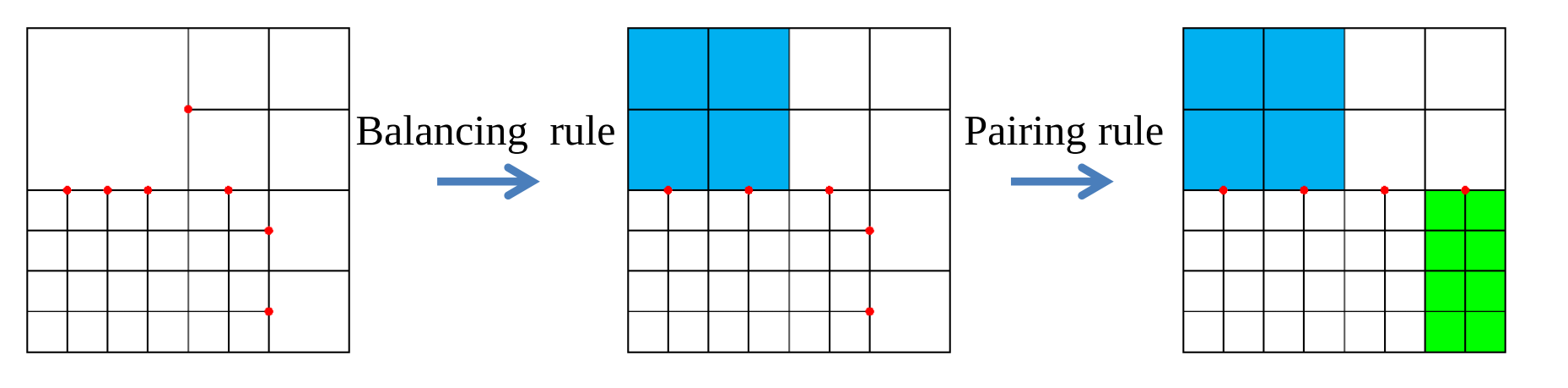


Fig. 2 Balancing rule (blue region) and pairing rule (green region).

Designing Templates and Constructing All-Hex Dual Mesh

The all-hex dual mesh contains no hanging nodes.

- Four transition cases for the cutting procedure in 2D.
- Only one transition template is needed (Fig. 3 (e)).

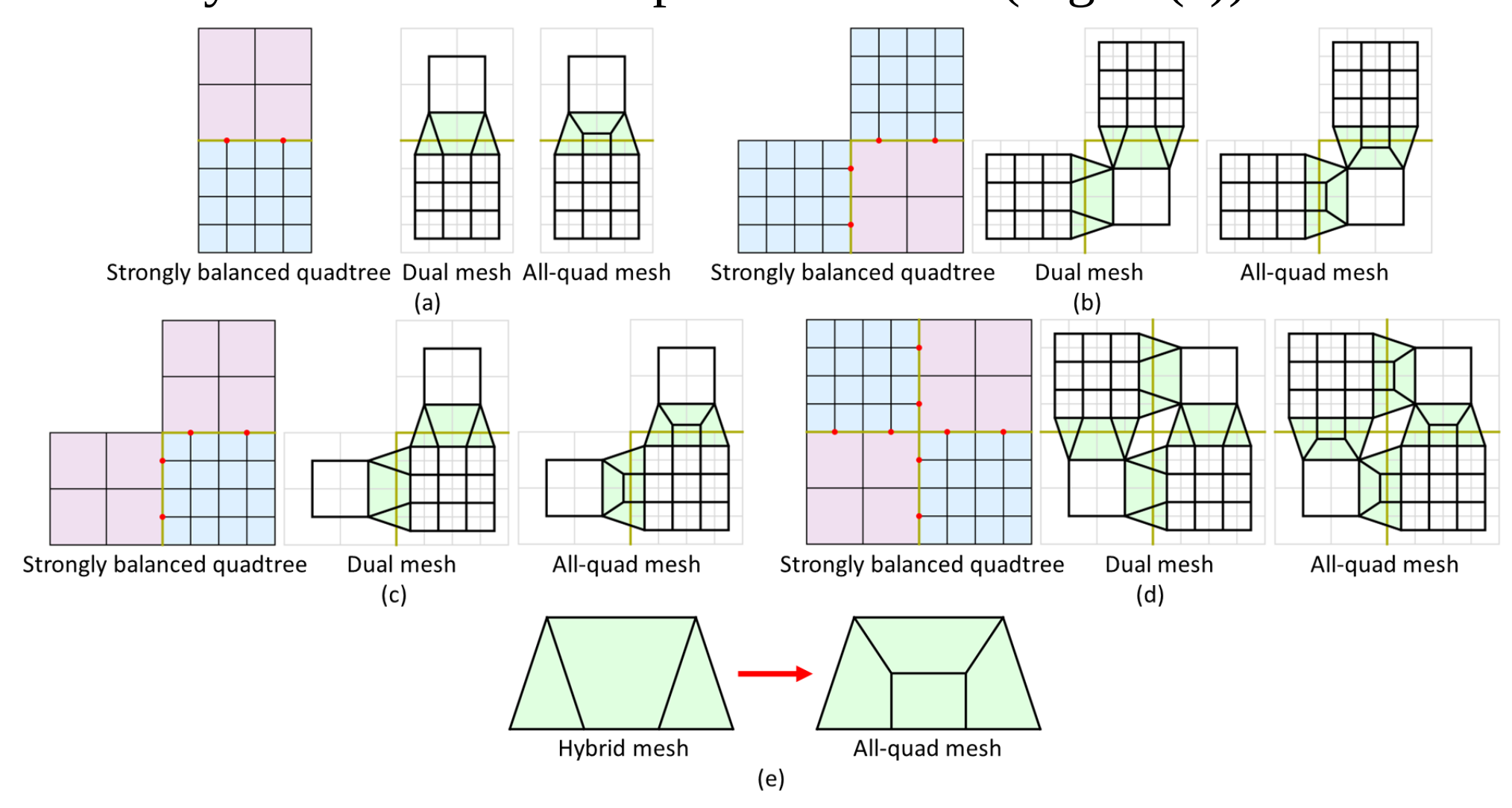


Fig. 3 Quadtree transition configurations (a-d) showing the strongly balanced quadtree, hybrid dual mesh, and the resulting all-quad mesh after applying the transformation template in (e). There are two red hanging nodes on each golden transition edge.

- Five transition cases for the cutting procedure in 3D

    One transition on face (Fig. 4 (a)).

    Four transitions on edge (Fig. 4 (b), (c), (d) and (e)).

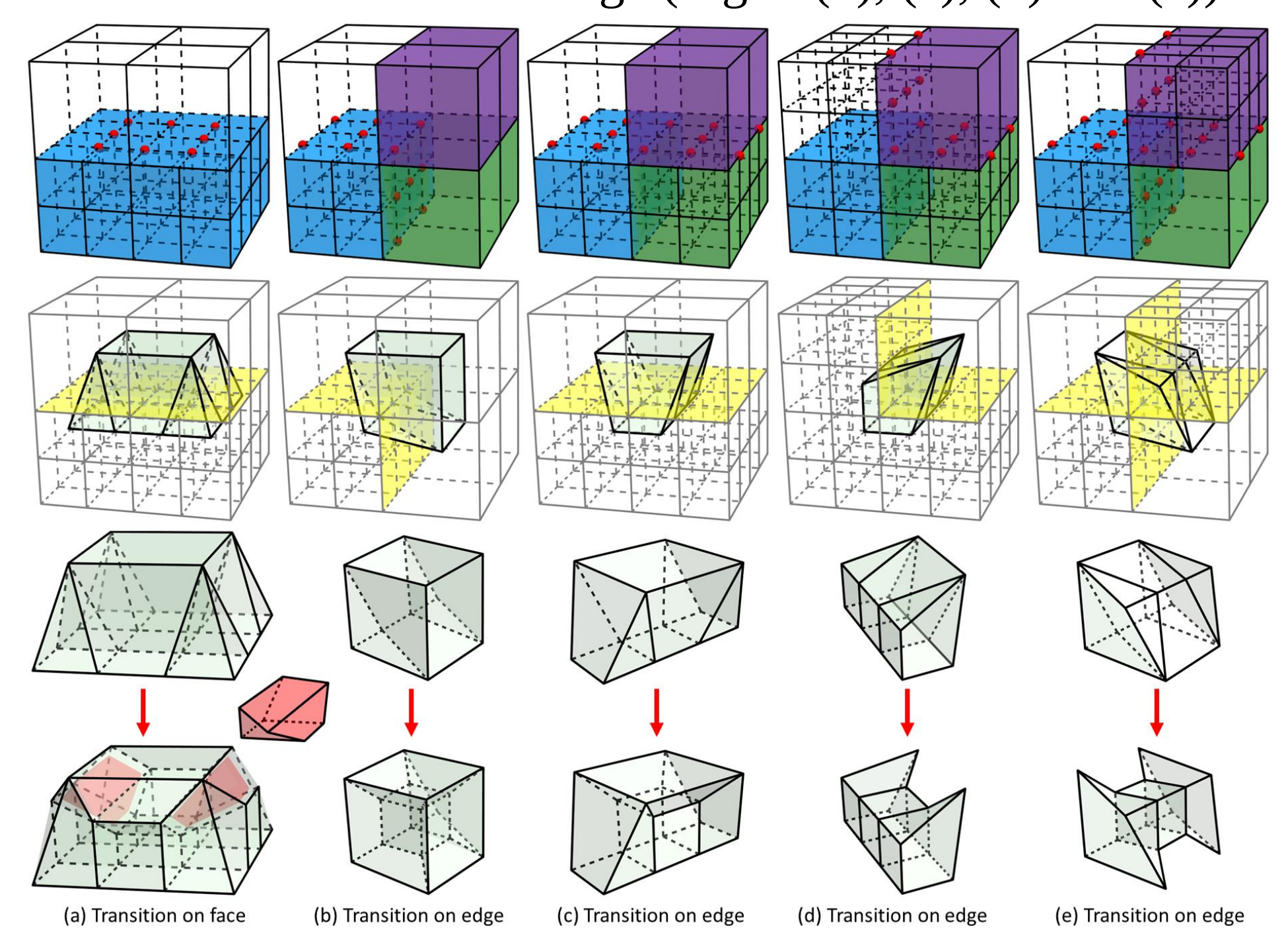


Fig. 4 Octree transition configurations (a-e) showing the strongly balanced octree (first row), hybrid dual mesh (second row), and the transformation template (third and fourth row).

We detect faces and edges in the strongly balanced octree and apply templates to generate an all-hex mesh, eliminating the need for a complex hybrid octree.

Clearing Buffer Zone

The elements outside and in the vicinity of the boundary surface are removed with coarse and fine criteria.

- **Coarse criterion**

    A hex element is removed if f_{min} + 0.1 * f_{max} < 0 is met. f_{min} and f_{max} represent the minimum and maximum signed distance functions of the corner points of the hex.

- **Fine criterion**

    Hexes attached to x are removed iteratively until $(n_i \times n_j) \cdot n_k > 0$ for all i, j, k = 0, 1, ..., m - 1 and i ≠ j ≠ k to prevent elements that will lead to poor scaled Jacobian elements in the buffer zone (blue and red in Fig. 5).

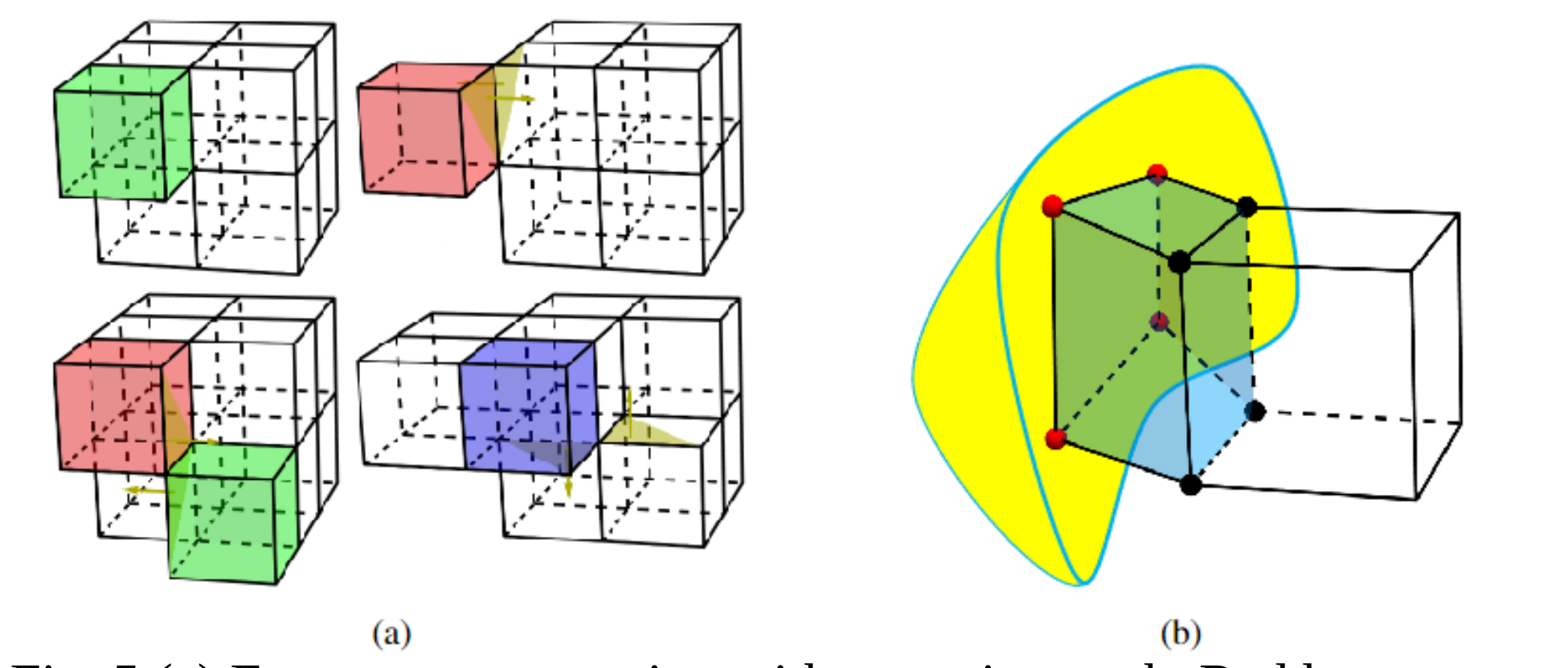


Fig. 5 (a) Four cases comparing with our prior work. Red hexes are removed by both methods, green only by the prior, and blue only by ours. Yellow triangles show normals violating fine criterion. (b) The buffer layer links core mesh boundary (black) to input surface (red).

- We found the non-manifold criterion inadequate for buffer zone element quality.
- Key factors are normals around core mesh boundary points.
- This approach is the key to enhancing the buffer zone element scaled Jacobian to over 0.5.

Quality Improvement with Jacobian Control

We mesh the buffer zone by connecting each boundary point x_i of the core mesh with its closest surface point x_i^s to form hex elements h_i, as shown in Fig. 5 (b).

Issue: The resulting elements in the buffer layer may be poor quality.

Goal: To preserve the positions of boundary points on the surface while relocating corners of worst-Jacobian elements to enhance the overall mesh quality.

Solution: Couple optimization with smart Laplacian smoothing.

Perform optimization to all the vertices in each iteration.

Perform smart Laplacian smoothing every 1, 000 iterations on the outmost two layers of vertices to smooth the surface and prevent getting stuck in local minima.

Detailed solution: The optimization is achieved by minimizing an energy function consisting of the geometry fitting, scaled Jacobian and Jacobian terms.

$$E = E_{GF} - E_{SJ} - E_J$$
$$= \sum_{i=0}^{nvert-1} \|x_i - x_i^s\|_2^2 - \sum_{i=0}^{m-1} \min SJ(h_i) - \sum_{i=0}^{n-1} \min J(h_i),$$

- nvert: the number of surface vertices
- x_i^s: closest surface point to x_i, updated every 1, 000 iterations for faster computation.
- m: the number of positive-Jacobian hexes
- n: the number of negative-Jacobian hexes.

We iteratively minimize E with gradient descent.

$$x_i \rightarrow x_i - \alpha \nabla E\Big|_{x_i}, i = 0, 1, \dots, 2 \times nvert - 1,$$

- $\alpha = 0.8 \times 10^{-3}$ for all tested models.

Optimization tricks:

- All surface points x_i are pulled onto x_i^s when the geometry fitting energy is small enough for exact surface reconstruction.
- The scaled Jacobian and Jacobian terms are only applied to hexes with SJ < threshold ε_{SJ} to prevent back-and-forth coordinate adjustment.
- Optimize with scaled Jacobian value when min $J(h_i) > 0$ to achieve optimal convergence. Optimizing with Jacobian instead cannot converge to the optimal position $|P_0P_1|$ or $|\nabla E|$ = 0 and will continue to $|P_0P_1|$ or $|\nabla E| \rightarrow -\infty$ (Fig. 6 (c, e)).
- Optimize with Jacobian instead of scaled Jacobian when scaled Jacobian is (1) in-differentiable or (2) negative to prevent gradient explosion (Fig. 6 (b)) or getting stuck in local minimum (Fig. 6 (d)).

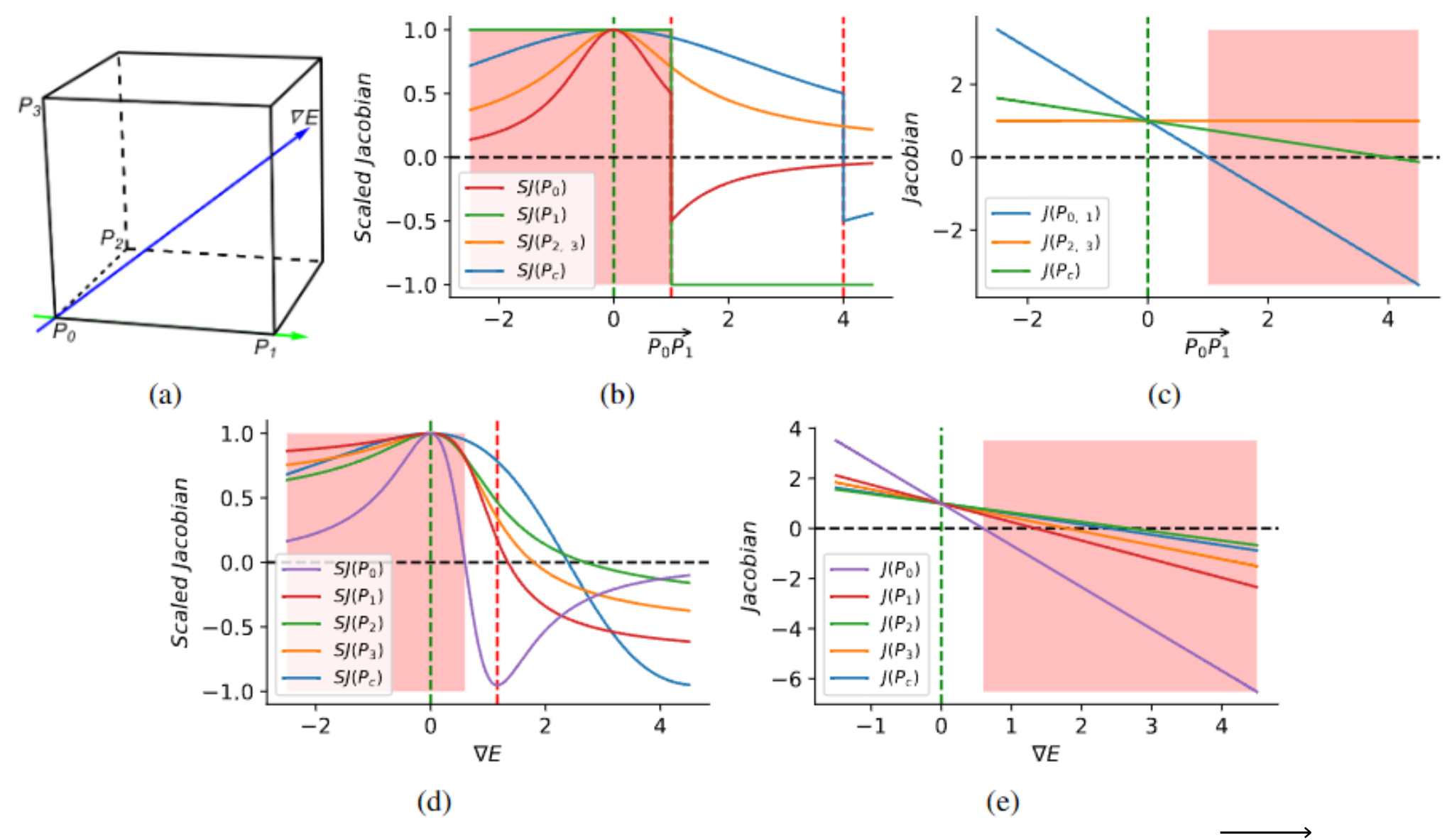


Fig. 5 (a) A hex with $|P_0P_1|$=1 is optimized by moving P_0 along $\overrightarrow{P_0P_1}$ in (b, c) or ∇E in (d, e). The optimum is at dashed green lines. Intervals for scaled Jacobian are red-shaded. In-differentiable points and valleys are dashed red lines. (b, d) show scaled Jacobian; (c, e) display Jacobian.

Differences compared to previous methods

- Previous methods:

    Removing non-manifold core mesh surface hexes.

    Applying geometric flow to boost overall mesh quality.

- Our approach:

    Removing core mesh surface hexes with the fine criterion.

    Using combined scaled Jacobian and Jacobian in gradient-based optimization to avoid local minima.

    Leading to notably high min scaled Jacobian > 0.5.

Numerical Results and Discussion

We run HybridOctree_Hex on a range of complex models without parameter adjustments. Twelve representative meshes are displayed in Fig. 6 and 7. Our results were computed on a PC equipped with a 3.6 GHz Intel i7-12700 CPU and 32GB of memory.

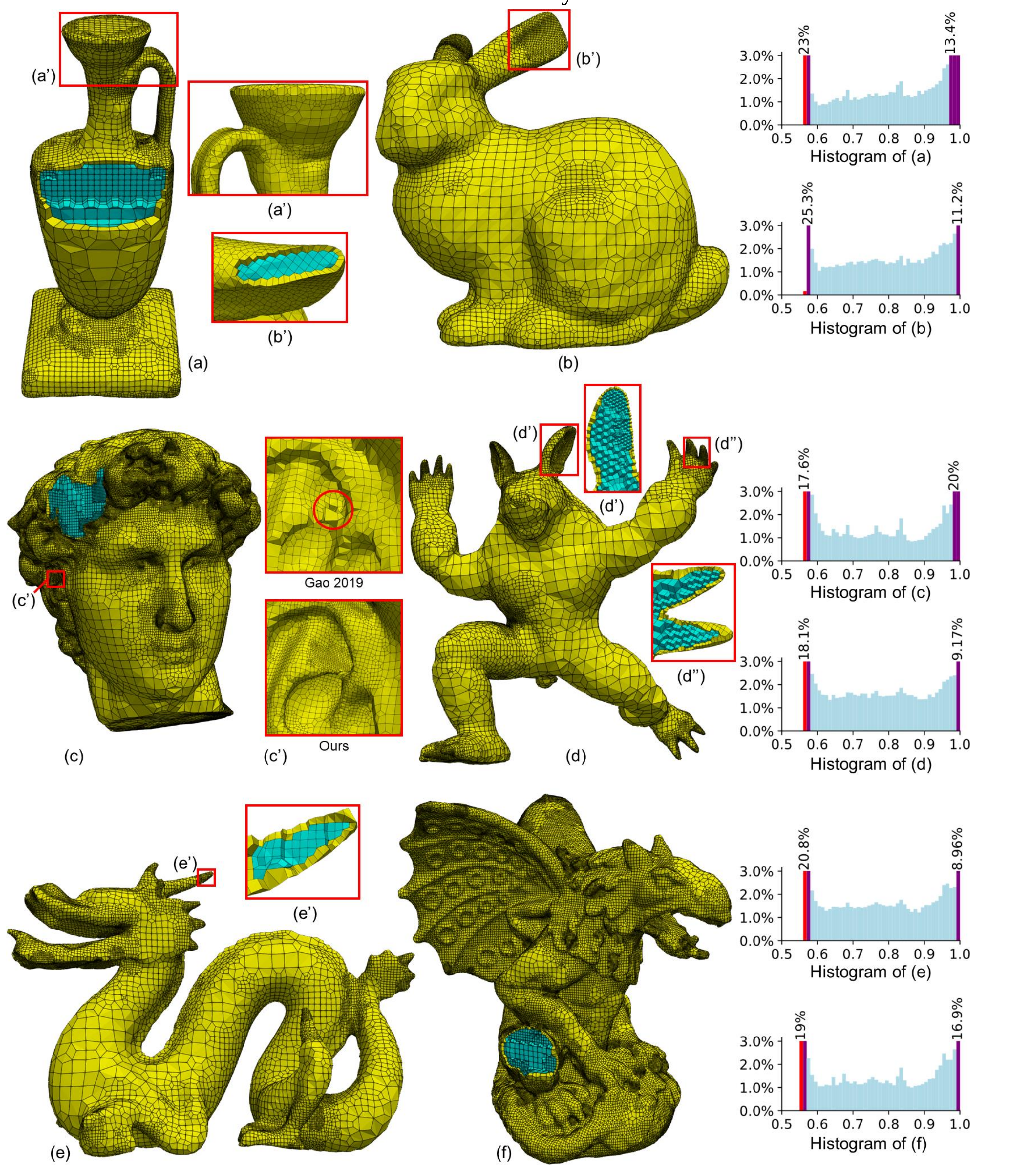


Fig. 6 (a) Bottle1; (b) Bunny; (c) David; (d) Deformed armadillo; (e) Dragon stand; and (f) Gargoyle. (a', b', c', d', d'', e') show zoomed-in images; (c') shows a comparison between our mesh and Gao 2019 mesh, which generates a hole at the thin region; the scaled Jacobian histograms are shown in the last column. The red bar represents the minimum scaled Jacobian and purple bars are truncated ones due to a higher frequency (≥ 3%).

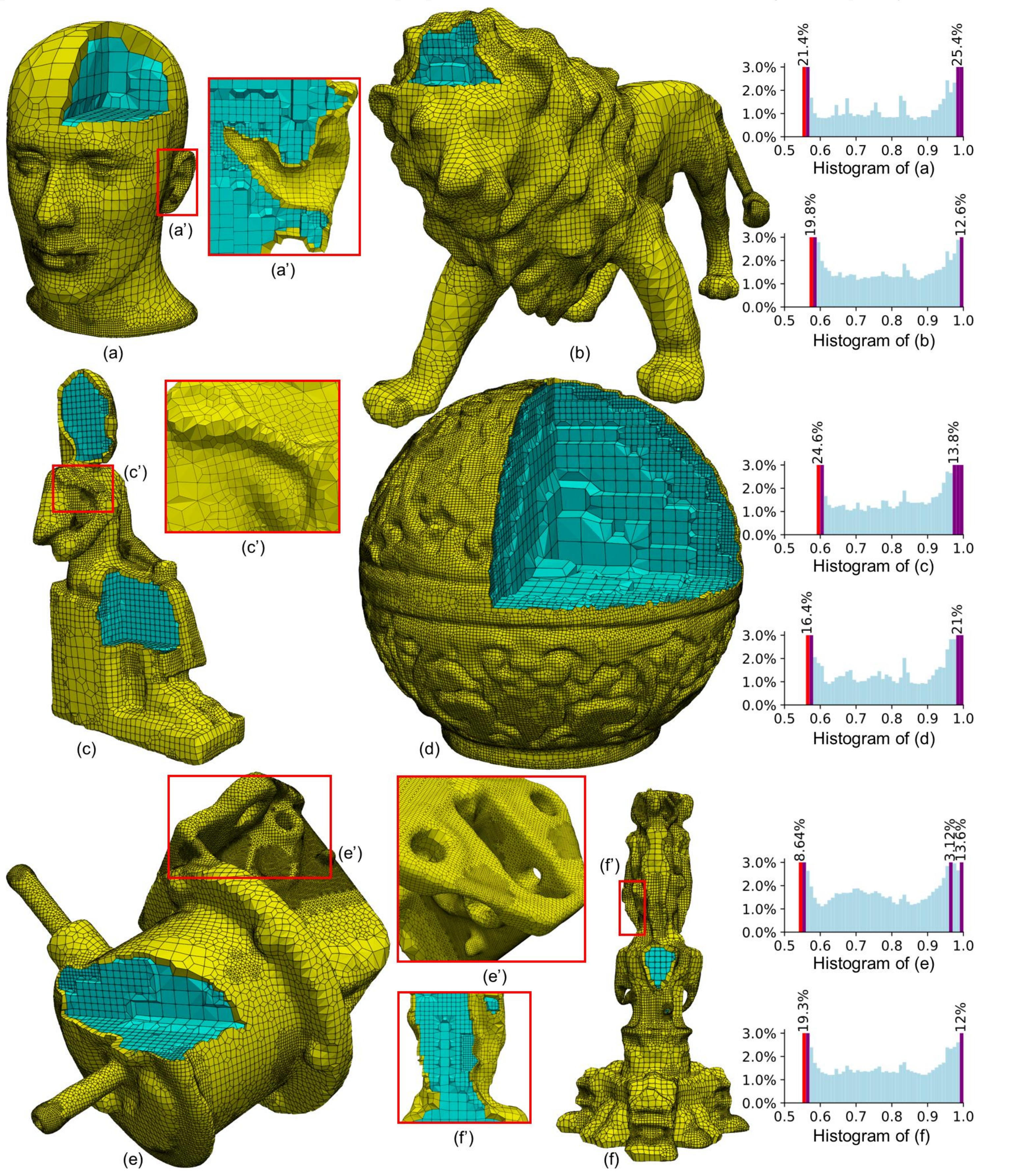


Fig. 7 (a) Head; (b) Lion recon; (c) Ramses; (d) Red circular box; (e) Oil pump; and (f) Thai statue. (a', c', e', f') show zoomed-in images; the scaled Jacobian histograms are shown in the last column. The red bar represents the minimum scaled Jacobian, and the purple bars are truncated ones due to a higher frequency (≥ 3%).

Tab. 1: Mesh time of all the tested models

Model	Time (s)	Model	Time (s)	Model	Time (s)
Bottle1	218	Bunny	358	David	10450
Deformed Armadillo	3, 431	Dragon Stand2	2, 052	Gargoyle	8, 769
Head	444	Lion Recon	2, 046	Oil Pump	9, 742
Ramses	713	Red Circular Box	7, 547	Thai Statue	1, 845

Conclusions

Contributions: We introduce HybridOctree_Hex, an advanced software for generating adaptive all-hex meshes. It efficiently detects surface features, constructs a balanced octree and an all-hex dual mesh. Our approach is robust, efficient, and automated on dozens of complex 3D models.

Limitations and future work: Future research can explore advanced techniques for feature detection and machine learning for octree level prediction. Customizing the energy function can further enhance mesh quality and feature preservation. Partitioning and parallel computing can allow detecting smaller features in a shorter time.

HybridOctree_Hex: Hybrid Octree-Based Adaptive All-Hexahedral Mesh Generation with Jacobian Control

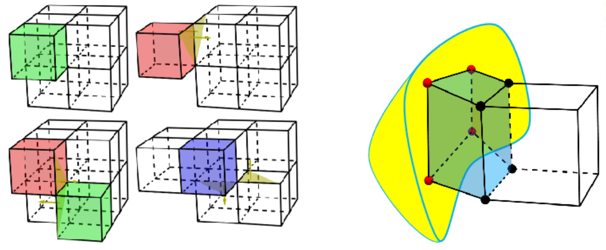
Hua Tong, Eni Halilaj, Yongjie Jessica Zhang
Computational Biomodeling Laboratory, Department of Mechanical Engineering, Carnegie Mellon University

(a) Initializing Octree

Refinement criteria:
(1) High curvature
(2) Small thickness

(d) Clearing Buffer Zone

Remove outside element with coarse and improved fine criteria.



(a) (b)

(e) Quality Improvement with Jacobian Control

(1) Optimize a new energy function with gradient descent

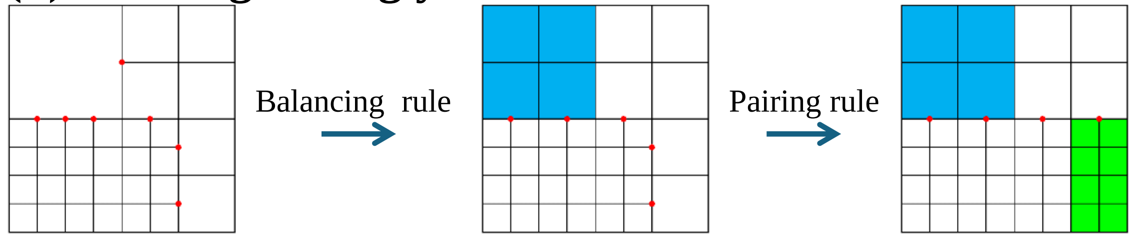
$$E = E_{GF} - E_{SJ} - E_J$$

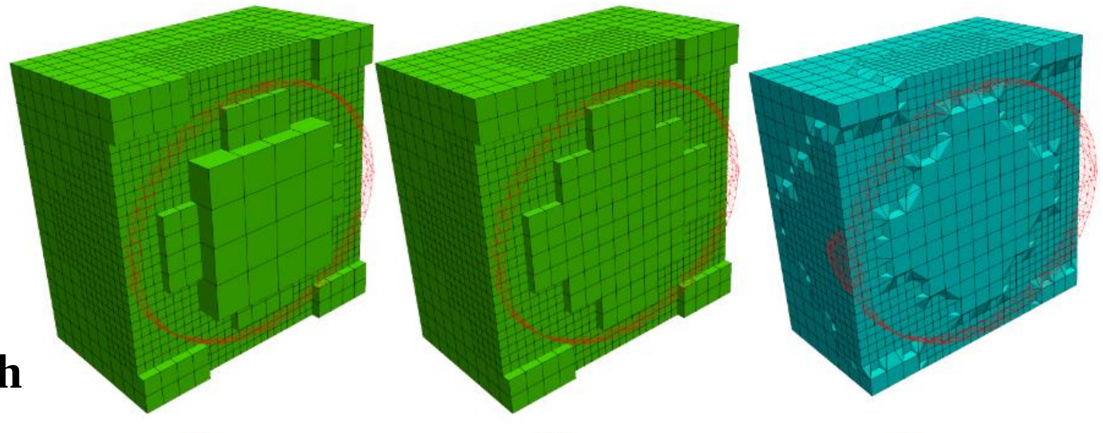
$$= \sum_{i=0}^{nvert-1} \|x_i - x_i^s\|_2^2 - \sum_{i=0}^{m-1} \min SJ(h_i) - \sum_{i=0}^{n-1} \min J(h_i)$$

(2) Couple with Laplacian smoothing to
Smooth the surface
Prevent getting stuck in local minima

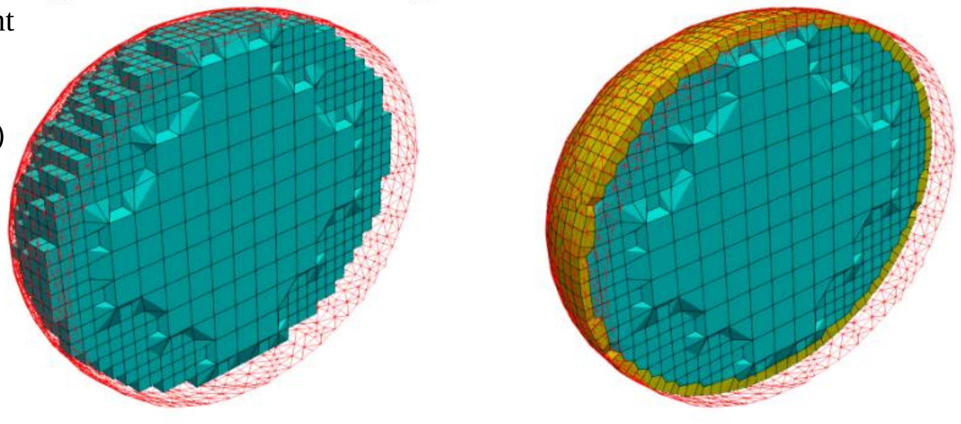
We obtain notably high minimum scaled Jacobian > 0.5

(b) Building Strongly Balanced Octree





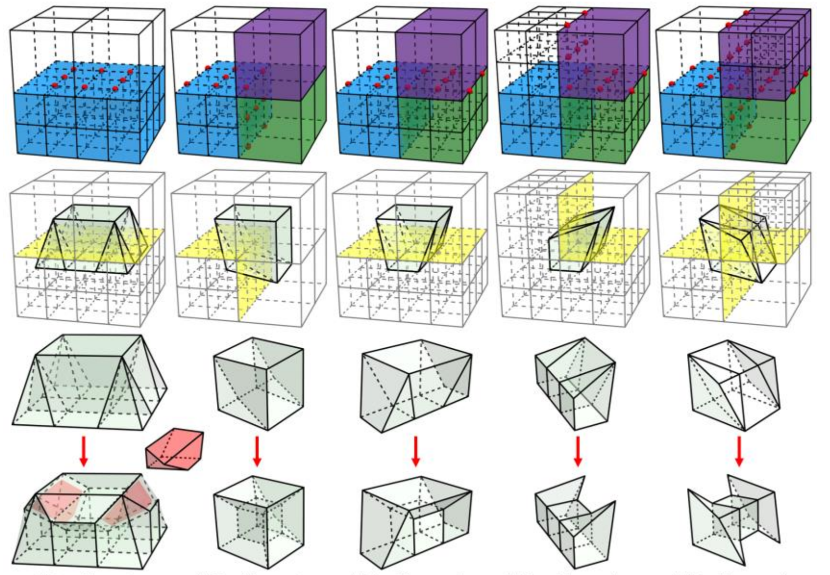
(a) (b) (c)



(d) (e)

(c) Constructing All-Hex Dual Mesh

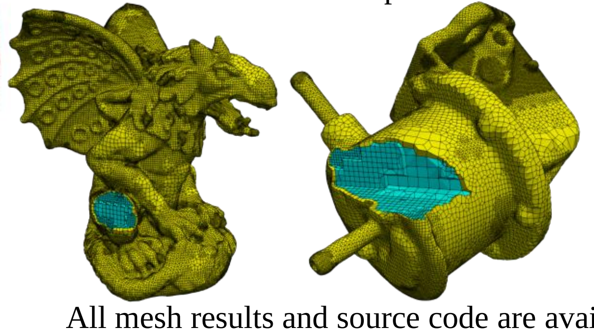
Five transition cases for the cutting procedure in 3D:
(1) One transition case on face
(2) Four transition cases on edge

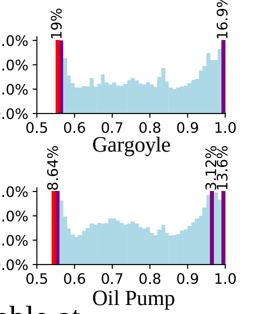


(a) Transition on face (b) Transition on edge (c) Transition on edge (d) Transition on edge (e) Transition on edge

(e) Numerical Results

Tested on dozens of complex models without intervention





All mesh results and source code are available at github.com/CMU-CBML/HybridOctree_Hex