Robust Structure Simplification for Hex Re-meshing

XIFENG GAO, New York University and University of Houston
DANIELE PANOZZO, New York University
WENPING WANG, The University of Hong Kong
ZHIGANG DENG, University of Houston
GUONING CHEN, University of Houston

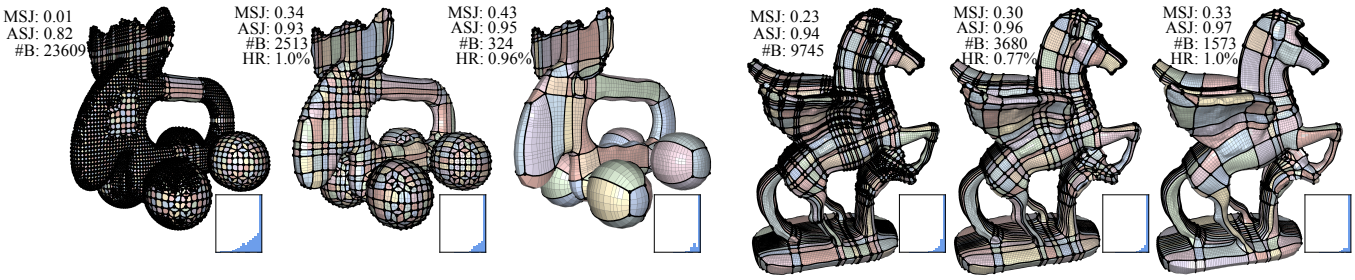


Fig. 1. When paired with an automatic octree method, our simplification algorithm can automatically and robustly produce coarse hexahedral meshes, starting from surface triangle meshes, which is tested on hundreds of models. Our method guarantees that all elements have positive scaled Jacobian and that the maximal deviation from the original shape is bounded by a user-specified Hausdorff distance. We show such an example in the teaser (left), where 98% of the structural components have been removed, with minimal changes to the boundary representation, and producing meshes with high average and minimum scaled Jacobians. Other meshing pipelines, such as [Fang et al. 2016], also benefit from our algorithm (right).

We introduce a robust and automatic algorithm to simplify the structure and reduce the singularities of a hexahedral mesh. Our algorithm interleaves simplification operations to collapse sheets and chords of the base complex of the input mesh with a geometric optimization, which improves the elements quality. All our operations are guaranteed not to introduce elements with negative Jacobians, ensuring that our algorithm always produces valid hex-meshes, and not to increase the Hausdorff distance from the original shape more than a user-defined threshold, ensuring a faithful approximation of the input geometry. Our algorithm can improve meshes produced with any existing hexahedral meshing algorithm — we demonstrate its effectiveness by processing a dataset of 194 hex-meshes created with octree-based, polycube-based, and field-aligned methods.

CCS Concepts: • **Computing methodologies → Volumetric models**;

Additional Key Words and Phrases: Hexahedral meshing, singularity structure, simplification, inversion-free.

ACM Reference format:
Xifeng Gao, Daniele Panozzo, Wenping Wang, Zhigang Deng, and Guoning Chen. 2017. Robust Structure Simplification for Hex Re-meshing. *ACM Trans. Graph.* 36, 6, Article 185 (November 2017), 13 pages.
DOI: 10.1145/3130800.3130848

This work was supported in part by the NSF CAREER award 1652515, NSF IIS-(1524782 and 1553329), NSFC-(61272019 and 61572292), and SZ-fund (SIRI0404201431).

DOI: 10.1145/3130800.3130848

1 INTRODUCTION

Hexahedral (or Hex-) meshes are one of the most commonly used volumetric discretizations for the computation of volumetric PDEs [Ramos and Simões 2006; Bourdin et al. 2007] due to their superior numerical properties and their natural support of tensor-product spline constructions [Li and Qin 2012]. In particular, hex-meshes are favored over tetrahedral meshes since simulations performed on hex-meshes can obtain a similar accuracy with a lower element budget, reducing both memory consumption and computation time [Benzley et al. 1995; Cifuentes and Kalbag 1992; Tadepalli et al. 2010]. In addition, hex-meshes with simple structures are favored by applications, like isogeometric analysis [Hughes et al. 2005; Bazilevs et al. 2006], in which basis functions with higher-order smoothness can be fitted to ensure a more accurate computation and faster convergence. Simple structures also facilitate the implementation of multi-grid solvers to accelerate the computation [Leonard et al. 2000; Wada et al. 2006].

Despite the tremendous research efforts in the last three decades, robust and automatic methods to produce valid, *coarse* hex-meshes with a simple structure are still elusive.

Octree-based methods [Maréchal 2009; Ito et al. 2009] can only create meshes with a high element count and complex structures. Polycube [Gregson et al. 2011; Livesu et al. 2013; Huang et al. 2014; Fang et al. 2016] and field-aligned [Li et al. 2012; Jiang et al. 2014] methods are not robust when used to create coarse meshes, leading to some inverted elements especially when the surface mesh is complex and contains high-resolution details. In practice, the only reliable way to obtain coarse meshes is the use of a user-assisted method, such as [Sandia 2016]. This hinders the applicability of hex-meshes, since a major manual effort is required in their generation.

We propose a simplification algorithm to reduce the topological (i.e., number of components and singularities) complexity of a hex-mesh, while guaranteeing: (1) a topologically valid output, (2) a positive scaled Jacobian for every element, and (3) the preservation of sharp features and a maximal, user-defined Hausdorff distance from the input surface. Topological validity is ensured by only performing simplification operations that do not add or remove handles, do not create non-manifold elements, and do not change the number of boundaries. Geometric validity is ensured by collapsing base complex sheets or chords using a locally-injective volumetric deformation enriched by a line-search strategy that prevents the solution from going outside of the valid space. Geometrical fidelity is ensured by only allowing operations that do not modify the boundary more than a user-specified distance threshold.

Our algorithm is a perfect complement for octree-based methods [Maréchal 2009; Ito et al. 2009], which are robust and automatic: when paired with our simplification algorithm, they lead to the first robust pipeline that can generate valid (i.e. with no flipped elements), and accurate hexahedral meshes with coarse structures for hundreds of models, without user-interactions (Figures 1 and 9). Other meshing pipelines based on polycubes or field-aligned parametrization can also benefit from our approach, as demonstrated in our experiments.

We provide our reference, open-source implementation in the additional material, in addition to all our results. The attached code can be used to reproduce all the results with standard parameters. The simplification sequences for all figures are attached as short mp4 movie clips.

2 RELATED WORK

We review the most relevant literature for the creation of hex-meshes. Many of these techniques are based on corresponding methods previously developed for quadrilateral meshing — we restrict our survey to volumetric meshing techniques, and we refer the interested readers to [Owen 1998] and [Bommes et al. 2013] for an overview of surface meshing techniques.

Paving and Sweeping. The first automatic hex-mesh generation techniques are paving (i.e., inserting regular layers of cubes aligned with a boundary quad mesh) and sweeping (i.e., extruding a partial quad mesh) [Shepherd and Johnson 2008; Gao et al. 2016]. While conceptually simple, they are extremely challenging to implement robustly, and they tend to introduce an excessive amount of singularities in the internal regions where the fronts meet (see Figure 12 for an example). Robust commercial implementations exist [Sandia 2016] and are currently widely used techniques, but often paired with other methods. The special case of tubular models has been considered in Livesu et al. [2016], where a skeleton is used to sweep a regular hex-mesh in its interior.

Spatial Partitioning. The intersection between a regular spatial subdivision and a mesh leads to a perfectly regular hex-mesh in the shape interior, with all the singularities concentrated on its boundary. Many methods have been proposed [Su et al. 2004; Zhang and Bajaj 2006; Zhang et al. 2007], and the most popular ones use adaptive grids (octrees) [Maréchal 2009; Ito et al. 2009; Zhang et al. 2013] to better approximate small features. The main limitation of these methods is that they concentrate all the singularities and low-quality elements on the shape boundary, which is usually the most interesting part in simulations. The upside is the high robustness of these methods, making them the default standard for automatic hex-mesh generation. Our algorithm enhances these methods, enabling them to robustly produce coarse meshes with a simple topology as we demonstrate in our test (Figure 9).

Polycube Deformation. Polycube methods [Gregson et al. 2011; Livesu et al. 2013; Huang et al. 2014; Fang et al. 2016; Fu et al. 2016; Li et al. 2013] deform a mesh into an axis-aligned polycube with edges of integer length. A canonical, axis-aligned pure hex-mesh can be created in the interior of the deformed mesh and warped using the inverse deformation inside the original shape. These methods better distribute the singularities on the boundary adapting them to the input geometry, but are restricted to polycube singularities and cannot introduce internal singularities, which are necessary to keep the element distortion low [Nieser et al. 2011]. These methods cannot guarantee to produce a valid polycube in the deformation phase and thus fall short of guaranteeing to produce a valid hex-mesh, limiting their practical applicability. Our method can further simplify the topology of valid meshes generated by these methods, as demonstrated in Figure 11.

Field-Aligned Parametrization. The most general and modern approaches to hex-meshing are field-aligned methods [Nieser et al. 2011; Huang et al. 2011; Li et al. 2012; Jiang et al. 2014]. A volumetric directional field [Vaxman et al. 2016] is used to guide a Poisson-based volumetric parametrization, which induces a hex-mesh if two conditions are satisfied: singularities should be placed at integer locations and the parametrization should be locally injective. Since these properties are not enforced by any of these methods, it is not always possible to use them to generate a valid mesh. Heuristic approaches that attempt to fix problems in the parametrization have been proposed [Lyon et al. 2016], but they can only amend minor problems and still fail for large and complex fold-overs. Generally, the success chances of these methods are proportional to the mesh resolution — the denser is the target mesh, the easier it is to parametrize, since the effect of the integer roundings becomes negligible [Bommes et al. 2009]. Our technique can be paired with field-aligned methods, allowing them to produce denser meshes, which are then coarsened by our collapse operations (Figure 10).

Simplification. Simplification methods interleave local and global coarsening operators to reduce the element count and singularities of existing quad meshes [Daniels et al. 2008, 2009a,b; Tarini et al. 2010; Bozzo et al. 2010]. Similar methods have been proposed for pure hex-meshes, using hexahedral sheets collapse operations [Borden et al. 2002; Ledoux and Shepherd 2010] or altering the hex-meshes via hexahedral duals [Tautges and Knoop 2003; Kowalski et al. 2012]. Unfortunately, neither method can guarantee to reduce the number of singularities and components in the base complexes (Figure 2). We propose an algorithm that builds upon the same operations, but that directly focuses on simplifying the global hexahedral structure and reducing the number of singularities. Also, we guarantee no flipped elements be introduced, which cannot be guaranteed previously.

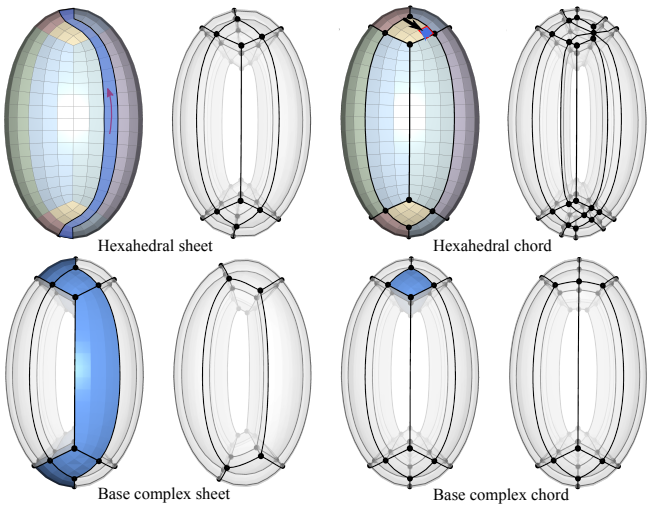


Fig. 2. Removing a hexahedral sheet (top, left) may not affect its structure layout, while collapsing a hexahedral chord (top, right) may even increase the complexity of the structure. In contrast, performing either a base complex sheet (bottom, left) or a chord (bottom, right) removal on the global structure of a hex-mesh will reduce the number of its components.

Singularity Alignment. A few techniques [Bommes et al. 2011; Tarini et al. 2011; Myles et al. 2010; Gao et al. 2015] have been proposed to reduce the complexity of the structure of a quad- or hex-mesh by aligning singularities and removing unnecessary spirals. However, without removing singularities, many meshes are still too complex to be used (as shown in Figure 12). In addition, none of them guarantees that no inverted elements are introduced during alignment. In contrast, our technique can efficiently reduce both singularities and components in the base complex without introducing any inverted elements.

Quality Improvement. Many approaches have been proposed to improve the geometric quality of hex-meshes while maintaining their connectivity fixed [Knupp 2000, 2003; Brewer et al. 2003; Wilson et al. 2012; Ruiz-Gironés et al. 2014, 2015; Livesu et al. 2015]. Mesquite [Brewer et al. 2003] is a popular method that has been used by many hex-meshing techniques [Gregson et al. 2011; Livesu et al. 2013; Huang et al. 2014; Fang et al. 2016] as a post-processing step, due to its efficiency and ease of use. Recently, Livesu et al. [2015] proposed an edge-cone optimization to untangle inverted elements, while also greatly improving the quality of the other elements. Both [Brewer et al. 2003] and [Livesu et al. 2015] strive to recover from inverted elements, but they are not guaranteed to succeed and might fail in challenging cases. Our method cannot recover from inverted elements, and it assumes an inversion-free mesh as input. The quality obtained by our method is comparable (and in some meshes superior) to these techniques, in particular in presence of sharp features (Figure 14).

3 STRUCTURAL SIMPLIFICATION

Our algorithm simplifies the base complex [Gao et al. 2015] of a valid hex-mesh, while guaranteeing to produce a mesh with a coarser structure and with positive scaled Jacobians [Stimpson et al. 2007].

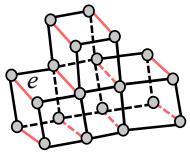
We extract *base complex sheets and chords* from the base complex of the input mesh (Section 3.1), and collapse them using a locally injective volumetric parametrization (Section 3.2). The collapse operations are sorted to accelerate the computation (Section 3.4). The top-ranked collapse operation is then checked to prevent it from violating topological invariants, deleting features, or changing the boundary more than a user-specified threshold on the Hausdorff distance (Section 3.3). Additionally, sheets and chords may be refined (introducing only regular elements) to maintain the desired number of hexahedral elements during the simplification. The algorithm iterates through these steps until the desired complexity is achieved, or whenever it cannot simplify the mesh further without violating the filtering criteria (Section 3.6).

3.1 Sheets and Chords

To keep the paper self-contained, we briefly introduce the notion of singularities, base complex, base complex sheets, and base complex chords (refer to [Gao et al. 2015] for more details). The latter two concepts are the extensions of the hexahedral sheets [Borden et al. 2002] and chords [Tautges and Knoop 2003] to the base complex.

Singularities. The singularities of a hex-mesh $\mathcal{H}$ are 1D curves, each of which is composed of a sequence of connected irregular hex edges with the same valence. The valence of a singularity is defined as the valence of the hex edges it contains.

Base Complex. From a singularity with valence n, n separation surfaces can be traced out and either meet with other singularities or terminate at the surface boundary. These separation surfaces and their intersections partition the volume domain $\mathcal{V}$ into topologically cube-like components, which provides a coarse structural tessellation of $\mathcal{V}$. We refer to the above structure as the *base complex* $\mathcal{B}$ of the hex-mesh $\mathcal{H}$. The base complex contains all the singularities of $\mathcal{H}$. A *component* of the base complex is a topological cube or torus. The faces of a component are collections of mesh faces, which we will refer to as *patches*.



Base Complex Sheet. We denote two base complex edges e_i and e_j in a patch as parallel ($e_i \parallel e_j$) if they do not share a vertex (Figure 3(a)). The *parallel edge set* of an edge e of a base complex is the set of edges that are parallel to e or to any other edge in the set. The red edges in the inset figure are a set of parallel edges. Note that in the rest of the paper, solid lines indicate visible lines, while dashed lines indicate occluded lines.

A base complex sheet is a collection of components that have one or more edges in the same parallel edge set. Its dual complex is a surface sheet composed of the dual patches (of the contained components) that are perpendicular to the parallel edge set defining this base complex sheet (Figure 4, middle). Figure 3 (middle) shows an example base complex sheet extracted from the base complex (left) of a bone mesh.

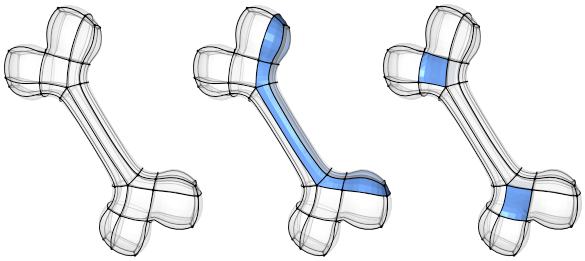


Fig. 3. The base complex (left), a sheet (middle) and two chords (right) of a bone mesh.

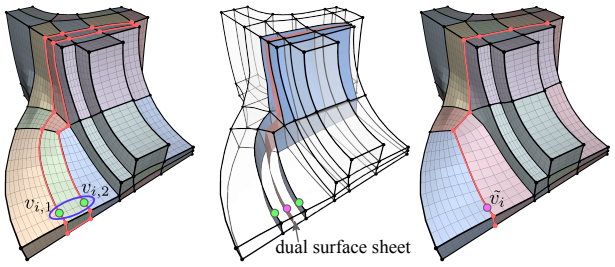
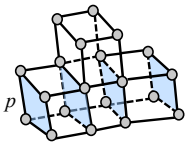


Fig. 4. A hex-mesh with a to-be-removed base complex sheet (highlighted by the red silhouette) (left); the dual surface sheet of the base complex sheet (middle), and the hex-mesh after collapsing (right). The green dots $v_{i,1}$ and $v_{i,2}$ are a pair of points in the i^{th} group that will be collapsed into a new point $\tilde{v}_i$ (purple) on the dual sheet.

Base Complex Chord. Similarly, two patches p_i and p_j belonging to the same component are parallel if they do not share a vertex. A *parallel patch set* of a patch p is the set of patches parallel to it or to any other patch in the set. The blue patches in the inset figure are a set of parallel patches.



A base complex chord is a collection of components that have one or more patches in the same parallel patch set. Figure 3(right) shows two example base complex chords.

For brevity, we refer to the base complex sheet (or chord) as sheet (or chord) in the rest of the paper. We show that by procedurally collapsing sheets and chords, the base complex $\mathcal{B}$ can be simplified, reducing the numbers of elements and singularities in the hex-mesh $\mathcal{H}$ (Propositions 3.1 – 3.4 in Appendix A).

3.2 Sheets/Chords Removal

We propose a parametrization approach to collapse chords and sheets. The interior of the hex-mesh is reparametrized onto itself to contract the sheet/chord that we wish to collapse to zero volume. This strategy is more expensive than previous methods that directly replace the sheet with its dual [Tautges and Knoop 2003] or combinatorially connect the two sides of the sheet and optimize the geometry afterward [Gao et al. 2015], but has the following major advantages: (1) it guarantees to produce a valid hex-mesh without inverted elements, (2) it distributes the distortion evenly over the optimization domain, and (3) it naturally supports constraints to preserve features.

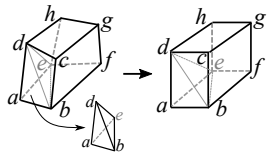
Without loss of generality, we focus on collapsing sheets and postpone the discussion of the chords to the end of this section.

Energy. Let $\mathcal{H} = (V, H)$ be the original hex-mesh, where V and H denote the sets of vertices and hexahedra, respectively. We want to compute the new positions of V such that (1) the space taken by the sheet has zero volume, (2) the scaled Jacobian computed at each hexahedron (which is not collapsed) is positive and optimized, and (3) the new surface boundary is as close as possible to the original one. We cast this problem as an energy minimization:

$$\min_V E(V) = E_D(V) + \lambda_t E_T(V) + \lambda_b E_B(V), \tag{1}$$

where $E_D(V)$ measures the geometric distortion of the hexahedra in H, $E_T(V)$ encourages the new positions of the vertices on the two side surfaces of the sheet to be on the dual surface sheet (Figure 4(b)), and $E_B(V)$ measures the deviation of boundary vertices from the original surface. The term $E_D(V)$ diverges if V contains an inverted element, infinitely penalizing any inversion. For all experiments, we use $\lambda_t = 10^5$ and $\lambda_b = 10^3$.

Distortion Term. A hexahedron has eight scaled Jacobians [Stimpson et al. 2007], each of which is computed on the tetrahedron defined by the three normalized outgoing edges from each of its eight corners (see the tetrahedron associated with corner a in the inset). We measure the distortion $D_h(V)$ of each hexahedron as the sum of the distortions of its eight tetrahedra $D_t(V)$.



$$E_D(V) = \sum_{h \in H} D_h(V) = \sum_{h \in H} \sum_{t \in h} M_t D_t(V) \tag{2}$$
$$D_t(V) = ||J_t||_F^2 + ||J_t^{-1}||_F^2$$

where M_t is the volume of a tetrahedron t and J_t is the Jacobian of the transformation of the tetrahedron t. After experimenting with all the energies proposed in [Rabinovich et al. 2017], we found that the best results, in terms of the quality of the elements, are obtained with the symmetric Dirichlet energy [Smith and Schaefer 2015]. Instead of encouraging each tetrahedron to be similar to a perfectly regular tetrahedron (as proposed in [Rabinovich et al. 2017]), we only require the ideal shape to be a right-angled tetrahedron, without constraining the length of its edges. Since a right-angled tetrahedron has three of its dihedral angles to be 90°, we compute the target shape as the tetrahedron of the corresponding corner of a cuboid, which has the same size and approximately the same edge ratios (see the inset). Specifically, the edge lengths in the three directions of the cuboid are determined and rescaled according to the average of the lengths of the four edges of the corresponding direction, i.e., we find the "ideal" cuboid, and encourage the mesh deformation algorithm to realize it.

We compare our proposal with competing hex-mesh optimization techniques in Section 4.

Zero Volume Term. The term $E_T(V)$ is a soft positional constraint that encourages the two sides of a sheet/chord to collapse into its

dual sheet (Figure 4(b)). We define $E_T(V)$ as:

$$E_T(V) = \sum_{i=1}^{K} \sum_{j=1}^{K_i} ||v_{i,j} - \tilde{v}_i||^2 \tag{3}$$

where K is the number of pairs (or groups) of collapsing points (Figure 4(a)), K_i is the number of points within a group. All the points in the i^{th} group (e.g., the green dots in Figure 4(a)) will be collapsed to a new point $\tilde{v}_i$ (e.g., the purple dot) on the dual surface sheet, as illustrated in Figure 4. $v_{i,j}$ denotes the j^{th} point in the i^{th} group. Note that if some of the points $v_{i,j}$ fall on the boundary surface, their opposite points will be collapsed onto them to preserve the boundary.

Boundary Preservation. The last energy term $E_B(V)$ allows vertices to slide over the surface, constraining them to lie on sharp features. Since we do not have manually annotated sharp features for all our models, we classify as sharp features all the edges whose dihedral angle is smaller than 140°degrees. We employ the strategy introduced in [Livesu et al. 2015], i.e., classifying all the boundary vertices into three types, i.e., corner vertices $v \in C$, feature line vertices $v \in L$, and regular vertices $v \in S$. These vertices are constrained to stay on their respective tangent planes, feature lines, and corners, respectively:

$$E_B(V) = \sum_{v \in C} ||v - \tilde{v}||^2 + \sum_{v \in L} ||v - \tilde{v} - a_l \vec{d}_l||^2 + \sum_{v \in S} ||\vec{n}(v - \tilde{v})||^2 \tag{4}$$

Here, $\tilde{v}$ is the closest surface position for each boundary vertex, $\vec{d}_l$ is the feature line tangent at $\tilde{v}$, a_l is an auxiliary variable added to the system for feature line constraints, and $\vec{n}$ is the normal at vertex $\tilde{v}$. Note that, our formulation is different from [Livesu et al. 2015] that minimizes a_l at the same time to prevent a vertex from going too far away from its original position, which may cause inverted quads on the surface. We let a vertex on a feature line freely slide along the feature curve to maximize the geometry quality while surface element flips are avoided by the flip-avoiding line search [Smith and Schaefer 2015].

Due to the linearization of the boundary constraints [Livesu et al. 2015], boundary vertices may leave their respective tangent planes or feature lines and move further away from their respective reference positions on the original surface. Therefore, after each deformation iteration, we update the surface correspondence information by projecting the vertices back onto the original surface or onto the features. Specifically, for a boundary vertex, we first re-compute its reference position on the original surface using Phong projection [Kobbelt et al. 1999; Panozzo et al. 2013]. Since we maintain the correspondences between the boundary vertices and the projected triangles of the original surface for all the collapsing and deformation steps, for each re-projection procedure of a boundary vertex, we can start a ring by ring, breadth-first-search (BFS) from its previous corresponding triangle, which in most cases will succeed without requiring a BFS. If the vertex is projected onto multiple triangles, we select the one with the minimal Euclidean distance and a consistent normal direction to its previous normal.

Solver. Our input is a valid hex-mesh (i.e., all scaled Jacobians are positive), and we use the SLIM solver [Rabinovich et al. 2017] to minimize the energy in Eq.(1), which uses a flip-avoiding line search [Smith and Schaefer 2015] to guarantee to produce a locally injective map. Since both the zero volume and the boundary term are added to SLIM as soft constraints, they compete with the distortion term, slowing down the optimization. We experimentally observed that SLIM is much faster when there are less soft constraints, and we thus divide the optimization into two stages: first collapse the sheet/chord (Eq.(1)), and then optimize for the distortion as described below.

Distortion Propagation. To distribute the distortion caused by the collapse operations, we solve the following optimization

$$\min_V E(V) = E_D(V) + \lambda_b E_B(V), \tag{5}$$

which is similar to the energy minimization of Eq.(1), but without the zero volume term. We use the same weight as Eq.(1).

Collapsing Sheets. A sheet is collapsed in two steps. We first minimize Eq. (1) to squeeze the volume of the sheet. A sheet is collapsed in two steps. We first minimize Eq. (1) to squeeze the volume of the sheet. If the sheet can be successfully collapsed (i.e., not introducing inverted elements), we then minimize Eq.(5) to distribute the distortion caused by Step 1. Otherwise, we continue to collapse the next sheet. Note that after each iteration of solving Eqs.(1) and (5), we update the boundary vertex correspondences using the aforementioned Phong projection to preserve surface features.

Collapsing Chords. A chord is the crossing of two locally perpendicular parameterization directions, each of which corresponds to a sheet. Collapsing a chord is achieved by moving one pair of opposite edges toward each other so that the chord has zero volume. Figure 5 illustrates this collapse operation in 2D. The two opposite nodes (red) merge into one so that the quad reduces to a curve (i.e. a diagonal of the original quad). This process requires the two pairs of opposite edges have the same number of samples. For instance, both pairs of the opposite edges in the left scenario in Figure 5 have two samples (gray dots). In this case, we can collapse the chord using the same deformation as for the sheets. Specifically, in the zero volume term $E_T(V)$, the target positions of the vertices on the side surfaces of the chord are vertices on the diagonal surface. When the two sheets have different numbers of samples (e.g., one pair of the opposite edges has three samples, while the other has two in Figure 5(right)), we first collapse a sub-sheet (the blue shaded sheet in Figure 5(right)) to make both sheets have the same number of samples, and then perform the chord collapsing as shown in Figure 5(left). Note that, since collapsing a sub-sheet is equivalent to shrinking the volume of the sheet where it is located, it will not affect the structure of the base complex.

Local Region. Note that in our implementation, in order to improve the computation performance, the above optimization is performed in a local region surrounding the selected sheet/chord instead of on the entire mesh. The local region consists of hexahedra within the β-ring neighborhood of the to-be-collapsed sheet/chord.

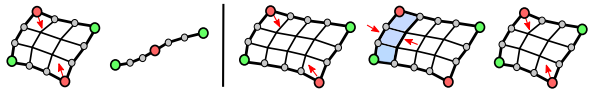


Fig. 5. 2D illustration of chord collapse. Two pairs of the opposite edges (left) have the same number of samples. The four edges collapse to the diagonal (i.e., the right figure in the left set of images). Two pairs of the opposite edges (right) do not have the same number of samples (i.e., one has two, while the other has three). A sub-sheet (blue) is then first collapsed before collapsing the chord.

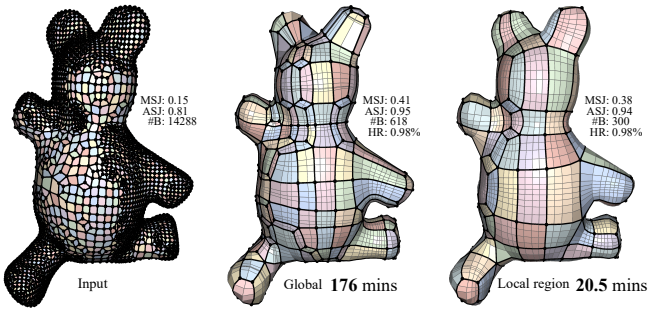


Fig. 6. A dense input mesh (left) is simplified restricting the quality optimization on a local region around each collapsed region (right), or by globally optimizing the quality over the entire mesh (middle). The local approach is 8 times faster than the global one and produces comparable results.

Using a small β will provide a massive performance boost, but decrease the quality of the mesh (Figure 6). After a number of experiments, we found that a good tradeoff between efficiency and quality is to set β to be four times the sheet/chord thickness.

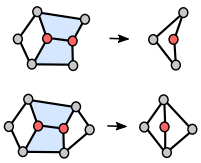
3.3 Filtering Criteria

Collapsing arbitrary sheets or chords might create non-manifold meshes, change the Euler characteristic, lead to the loss of sharp features, or introduce singularities of low valences (e.g. valence-1 and valence-2). To avoid these problematic cases, we simulate each collapse operation and discard those that violate any of the following validity criteria.

Euler Characteristics. During the simplification, we maintain the Euler characteristics (#V+#F-#E) for both the closed surface of the hex-mesh and its corresponding base complex. A common case prevented by this criterion is the filling of small holes.

Manifoldness. Removing a sheet (Figure 7(left)) or a chord (Figure 7(right)) may lead to non-manifold meshes, such as two components on the surface sharing either a node or an edge. We prevent these cases by only allowing operations that preserve manifoldness.

Edge Valence. During a collapse operation, many pairs of edges are collapsed into single edges. Let e_1 and e_2 be a pair of edges that are collapsed into a single edge e'. In normal situations, the valence (i.e. the number of incident hexahedra)



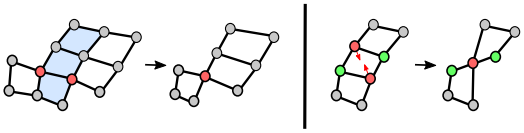


Fig. 7. Removing a sheet (left) by collapsing the left and right surfaces results in a non-manifold configuration where two components share an edge; while collapsing the chord (right) leads to another non-manifold configuration where two components share an edge.

of e' is higher or equal to the minimum of the valence of e_1 and e_2. However, there are degenerate cases, where the collapse can potentially generate nodes with lower valences. The inset figure provides examples of 2D valence-1 (top) and valence-2 (bottom) vertex configurations, which correspond to valence-1 and valence-2 edges in 3D, respectively. While these cases are topologically valid, they are undesirable since they lead to elements with negative Jacobians. We thus prevent all the collapses of pairs of edges for which:

$$\tau_{e'} < \min(\tau_{e_1}, \tau_{e_2})$$

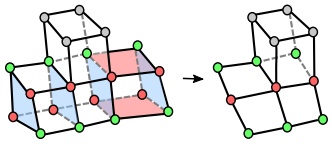
where τ_e denotes the valence of e. To enforce this criterion, the valences of the edges after collapsing need to be computed.

Predicting Edge Valence. The valence of a new edge e' created by a collapse can be predicted without actually collapsing the sheet or chord as

$$C'(e') = (C(e_0) - Cr(e_0)) \cup (C(e_1) - Cr(e_1)) \cup \ldots \cup (C(e_n) - Cr(e_n)) \quad (6)$$

where $C(e)$ is the set of the neighboring components of e in the base complex before simplification, $Cr(e)$ is the set of its neighboring components being removed, and edge e' is newly created due to the collapsing of a group of edges $\{e_0, e_1, \ldots, e_n\}$. See Figure 4 (left) for an example. Note that $\tilde{v}_i$ is one end point of the edge e'. A more detailed explanation and some examples on the valence prediction are provided in the Appendix.

Boundary Conformity. During simplification, the surface of the hex-mesh deviates from the original boundary either when removing a sheet or a chord



that represents protruding features of the model or when boundary vertices move far away from the original surface during the optimization. The former case happens when removing a sheet (or chord) erases other sheets (chords), as illustrated in the inset. Removing the chord represented by the blue parallel patches would lead to the simultaneous collapse of the chord represented by the red parallel patches. The surface parts of these sheets or chords (e.g., the chord represented by the red patches) typically represent protruding features of the model. To preserve features, we disallow the removal of a sheet or a chord that would lead to the complete removal of other sheets and chords. In particular, we prevent the collapse of a sheet or a chord, if the two vertices that are about to be collapsed into one (Figure 4) have one of the following three conditions: (1) both of them are classified as corners (see the boundary preservation in Section 3.2); (2) they are located on two different surface

feature curves, and (3) they are on two different boundary patches. To prevent the boundary vertices from moving too far from the original surface, we use Metro [Cignoni et al. 1996] to compute the Hausdorff distance between the resulting surface and the original surface after each removal. If the ratio of the maximum Hausdorff distance to the bounding boxes of both meshes is larger than a user-specified threshold (we used 1% in all our experiments), we disallow this removal. To reduce unnecessary computation, the Hausdorff distance is only computed if the collapse operation satisfies all the other criteria.

3.4 Ranking

Any sequence of collapses that satisfies the above filtering criteria (Section 3.3) will lead to an inversion-free and manifold mesh with the same Euler characteristic as the input. However, randomly selecting sheets/chords to collapse may require many queries to find an appropriate candidate without violating the filtering criteria.

To accelerate the selection of the appropriate sheets/chords to collapse, we propose a simple yet effective heuristic to prioritize sheets/chords according to the width of the sheets/chords. Let w be the average length of all the parallel edges in the sheet (for a chord it is the average length of the diagonal of all the parallel patches in the chord). The smaller the w is, the higher the corresponding sheet/chord is ranked and it will thus be tested first as a candidate for the collapse operation. The reasoning behind this heuristic is that collapsing a skinny sheet/chord has a higher chance to satisfy the filtering conditions than a thick one, since the former requires a smaller deformation. Figure 8 compares the result obtained with random order (middle) and the one with our proposed width ranking (right). While the two results are comparable (with a slight edge in simplicity for the one on the right), the running time is drastically reduced with our heuristic, that reduces it from 23.5 to 5.5 minutes.

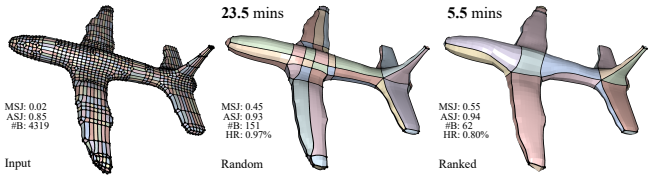


Fig. 8. A dense hex-mesh (left) is simplified by our algorithm without ranking the simplification operations (middle), and using our heuristic ranking based on the width of the sheets/chords (right). Note the major reduction in the running time, from 23.5 to only 5.5 minutes.

3.5 Sheet Refinement

Note that in the above pipeline, after the collapsing of a sheet, a sheet refinement process is performed to attain the same or similar number of hex elements to a target number (e.g., the number of elements in the input mesh or a user input number). We achieve this as follows. We first rank the sheets based on the length of hex-edges contained in the parallel edges of the base complex. The higher the average, the higher the priority given. We then split every hexahedron of the top-ranked sheets into two in the direction perpendicular to the parallel edges of the sheets. Since the number of elements will be doubled within the sheet, we can estimate how many sheets need to be subdivided in order to achieve the target number of elements. We perform this operation only if the resulting elements are not inverted.

3.6 Summary

Given an all-hex mesh as the input, we extract its base complex and simplify it by applying valid collapse operations, sorted by their thicknesses. At a high-level, our algorithm iterates the following steps until a termination criterion is reached. We use the Hausdorff distance ratio r_h as the termination condition in all our experiments. The user also specifies the desired element number as the ratio w.r.t the input element number, $r_{|H|}$.

(1) Extract all base complex sheets and chords from the base complex (Section 3.1) and prioritize them according to their thicknesses w (Section 3.4);
(2) Find a top-ranked valid sheet/chord and predict the edge valence;
(3) Check the filtering criteria (except the Hausdorff distance ratio) (Section 3.3). If satisfied, perform the collapse operation; otherwise, move to the next sheet/chord;
(4) Remove the sheet/chord using the locally injective volumetric reparametrization (Section 3.2). If a valid parameterization is not found (i.e. inverted elements are introduced) or the Hausdorff distance ratio after the collapse is larger than r_h (Section 3.3), this operation is reversed and the procedure (2-4) is repeated for the next sheet/chord until a successful operation is performed;
(5) Perform sheet refinement on the hex-mesh (Section 3.5) if the number of its contained hexahedra is below the user-specified threshold.

Finally, we perform a global optimization to improve the quality of the elements by minimizing Eq. 5.

4 EXPERIMENTS

We tested our simplification algorithm on a six-core i7 processor clocked at 3.5 Ghz with 64 GB of memory, using a single thread implementation. We limit the maximal number of iterations of the SLIM solver to 5, which achieves a good tradeoff between quality and computational cost. We use the same set of parameters ($r_h = 1\%$ and $r_{|H|} = 1.0$) for all experiments and figures — the exceptions are listed in the corresponding figure captions.

Quantitative Evaluation. Table 1 provides the statistics of a subset of our tested meshes (the full database of 194 models and their statistics are provided in the supplemental material). For both the input and output meshes, we report the numbers of the hex elements (#H), the numbers of base complex components (#B), the scaled Jacobians in the format of minimum/average/standard deviation (MSJ/ASJ/Std.). For each output mesh, we also provide the Hausdorff distance ratio (HR) computed as the maximum Hausdorff distance to the input mesh w.r.t. the bounding boxes of both the input and output meshes, and the component reduction ratio (R) computed as $(\#B_{in} - \#B_{out})/\#B_{in}$. HR is defined as $\max[d(S_i, S_o), d(S_o, S_i)]/D_{box}(S_i \cup S_o)$, where S_i and S_o are the surface of the input and output hex-meshes, $d(X, Y)$ measures the Hausdorff distance [Cignoni et al.

1996], $D_{box}(X)$ is the diagonal of the bounding box of X, respectively. The timing information for the simplification of each model is also reported. From these results, we see that most of the meshes achieve significant simplification of their mesh structures (see the column of **R**), and their Jacobian quality has also been improved (see the decrease of the Std. for all models). The running time over the entire dataset ranges from 0.03 minutes (#H = 2178 and #B = 7) to 533 minutes (#H = 72041 and #B = 72041). The average running time is 57 minutes. The running time for each model is included in the additional material.

Octree-based Methods. Octree-based methods are ideal to generate the input meshes for our algorithm, since they can robustly and automatically mesh complex surface models. To demonstrate the robustness of our pipeline, we used [MeshGems 2015] to mesh all the 106 models in the [Fu et al. 2016] database, which are then simplified by our algorithm (Figure 9). On average over the entire database, we obtained 86% complexity reduction, and 1100%/13% MSJ/ASJ improvements over the inputs, while maintaining a 0.92% Hausdorff distance ratio.

Fig. 9. Simplified results (right) on meshes (left) generated with the octree-based method [MeshGems 2015].

Frame-Field Methods. Our algorithm can simplify models created by frame-field methods. We experimented with all the 15 models provided by [Li et al. 2012] (Figure 10). Since our method only collapses chords and sheets that satisfy the topological constraints, the surface features are nicely preserved in our outputs. On average over all these meshes, our simplification algorithm achieved 60% complexity reduction, and 1.2%/3% MSJ/ASJ improvements over the inputs, while maintaining a 0.6% Hausdorff distance ratio.

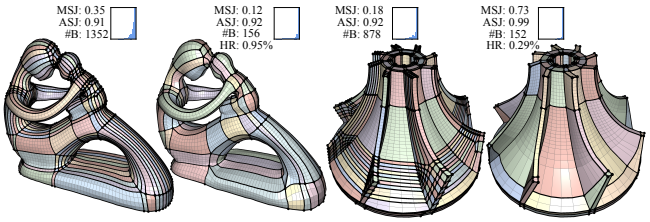


Fig. 10. Simplified results (right) on meshes (left) generated with the frame-field based method [Li et al. 2012].

Polycube Methods. Our algorithm can also simplify the meshes created by polycube methods. In Figure 11, we show the results of the simplification, starting from meshes created by a variety of polycube based methods [Gregson et al. 2011; Fang et al. 2016; Fu et al. 2016]. Results for all the 73 polycube hex-meshes including those obtained from other polycube methods (i.e. polycut [Livesu et al. 2013], polycube simplification [Cherchi et al. 2016] and $l1$−polycube [Huang et al. 2014]) can be found in the supplemental materials. On average over all these meshes, our simplification algorithm achieved 59% complexity reduction, and 6%/4.3% MSJ/ASJ improvements over the inputs, while maintaining a 0.37% Hausdorff distance ratio.

Comparison with [Gao et al. 2015]. The alignment method proposed in [Gao et al. 2015] does not remove singularities, drastically limiting the valid removal operations (Figure 12). In contrast, our method produced simpler structures for all the models used in [Gao et al. 2015] except the sculpture-A model (the complete statistics are in the supplementary material). For the sculpture-A, the alignment method reduces the number of components to 16 (Figure 13 (left)), while our method with $r_h = 1\%$ reduces it to 23 (Figure 13 (middle)). However, by setting $r_h = 3\%$ we achieve better reduction ratio (Figure 13 (right)) with comparable mesh quality. More importantly, our method always produces elements with positive Jacobian, while Gao et al.'s method [2015] does not guarantee it. We have applied Gao et al.'s method to our entire dataset of 194 models and found that it fails to produce a valid, inversion-free mesh for 91 of them (47%). For instance, the isidore model shown in Figure 12 has inverted elements after performing Gao et al.'s simplification. Table 2 provides the statistics of the geometric quality, structure simplicity, and the timings for the models compared in Figure 12, which shows that our algorithm has comparable computational efficiency to [Gao et al. 2015].

Comparison with [Brewer et al. 2003] and [Livesu et al. 2015]. We show the superiority of our proposed distortion minimization (Eq.(2)) with the popular Mesquite [Brewer et al. 2003] and the state-of-the-art geometric optimization by Livesu et al. [2015] on all the 194 models. For each model, we optimize its geometric quality without simplifying its structure by minimizing the distortion propagation energy (Eq.(5)). The optimization terminates when Hausdorff

Table 1. Statistics of the input and output meshes of the representative models shown in the paper. #H stands for the hex element number, #B is current number of base complex components, ASJ and MSJ are the average and minimum scaled Jacobian. Std. is the standard deviation of the distribution of scaled Jacobians (in percentage) of all elements in a mesh. HR indicates the Hausdorff distance, and R is the percentage of reduction. Please refer to the supplemental document for the statistics of all models we have experimented with.

Models	Input			Output					
	#H	#B	MSJ/ASJ/Std.	#H	#B	MSJ/ASJ/Std.	HR (%)	R	Time (m)
airplane1 (Figure 9)	4972	3661	0.03/0.84/2.45	4685	200	0.45/0.93/ 0.84	0.89	95%	10.2
airplane2 (Figure 8)	6118	4319	0.02/0.85/2.49	6154	62	0.55/0.94/0.65	0.80	99%	11.0
chair1 (Figure 9)	11686	9322	0.02/0.84/2.03	10314	560	0.29/0.91/0.84	0.97	94%	46.1
cup (Figure 9)	48341	48075	0.03/0.87/1.96	40144	382	0.45/0.96/0.41	0.90	99%	227.0
deckel (Figure 9)	53658	53116	0.03/0.84/3.50	48822	2122	0.24/0.94/0.96	1.0	96%	504.1
elk (Figure 1)	24916	23609	0.013/0.82/3.49	15156	324	0.43/0.95/0.58	0.96	98%	87.0
kiss (Figure 9)	18418	18360	0.03/ 0.84/2.27	15492	1174	0.44/0.94/0.68	1.0	94%	47.6
toy1 (Figure 9)	18947	18883	0.12/0.81/2.60	12972	481	0.42/0.94/0.61	0.89	97%	32.9
fertility (Figure 10)	13584	1352	0.35/0.91/0.77	8060	156	0.12/0.92/2.24	0.95	88%	11.9
impeller (Figure 10)	11174	878	0.18/0.92/1.07	11496	152	0.73/0.99/0.1	0.29	83%	5.7
armadillo (Figure 11)	78376	5960	0.27/0.91/0.64	57567	549	0.29/0.97/0.32	0.99	91%	295.3
bunny (Figure 11)	81637	1324	0.14/0.93/0.71	71440	153	0.43/0.98/0.21	0.93	88%	110.7
carter (Figure 11)	64911	1101	0.22/0.94/0.78	51100	141	0.28/0.97/0.38	0.98	87%	38.7
dragon (Figure 11)	114178	12488	0.16/0.92/0.74	89348	1927	0.28/0.96/0.33	1.0	85%	348.2
pegasus (Figure 1)	120344	9745	0.23/0.94/0.55	94492	1573	0.33/0.97/0.34	1.0	84%	316.4

Table 2. Comparison with [Gao et al. 2015]. #H stands for the hex element number, #B is current number of base complex components, ASJ and MSJ are the average and minimum scaled Jacobian. Std. is the standard deviation of the scaled Jacobians (in percentage) of all elements in a mesh. HR indicates the Hausdorff distance, and R is the percentage of reduction.

Models	Input			[Gao et al. 2015]						Ours					
	#H	#B	MSJ/ASJ/Std.	#H	#B	MSJ/ASJ/Std.	HR (%)	R	Time (m)	#H	#B	MSJ/ASJ/Std	HR (%)	R	Time (m)
3torus	4654	3706	0.33/0.87/1.47	6586	3033	0.29/0.89/1.29	0.53	18%	14	4208	71	0.62/0.94/0.47	0.82	98%	8
isidore	21695	21679	0.02/0.83/2.73	21542	21526	-0.11/0.83/2.80	0.26	0.7%	85	13744	1103	0.36/0.95/0.66	1.0	95%	45
joint	17784	83	0.73/0.98/0.07	15450	59	0.74/0.98/0.05	0.17	29%	12	19422	17	0.66/0.99/0.05	0.10	80%	7
angel_1	14068	1284	0.47/0.92/0.61	14728	698	0.30/0.92/0.72	0.53	47%	18	11891	211	0.52/0.96/0.40	0.99	84%	24
dancing children	35293	5482	0.14/0.87/1.13	37978	1458	0.19/0.88/1.17	1.09	73%	39	30121	998	0.30/0.95/0.50	1.0	82%	65
gargoyle	25669	7563	0.20/0.91/0.82	28368	1920	0.21/0.91/0.87	1.18	75%	42	22524	805	0.14/0.96/0.46	0.98	89%	30

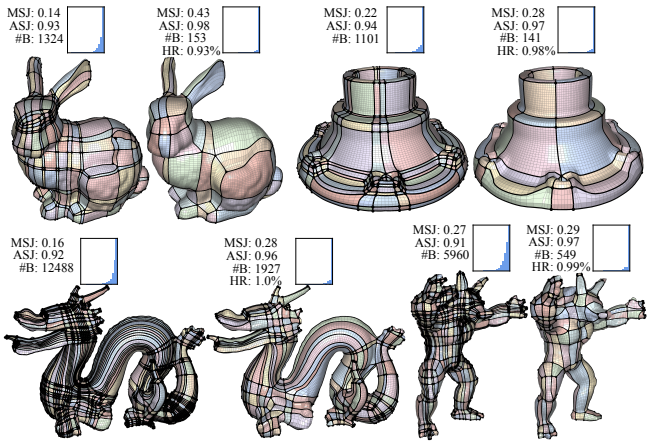


Fig. 11. Simplified results (right) on meshes (left) generated with a variety of polycube approaches, including bunny by volumetric polycube [Gregson et al. 2011], armadillo by efficient polycube [Fu et al. 2016], carter and dragon by [Fang et al. 2016], respectively.

distance ratio is larger than 1% or the MSJ is not increasing anymore, or after a maximum 10 iterations has passed. On average, we achieved 0.21/0.04, 0.16/0.03, and 0.02/0.01 higher MSJ/ASJ over the input, the results from Mesquite and Livesu et al. [2015], respectively. Since both Livesu et al. [2015] and our optimization moves boundary vertices, we compared the Hausdorff distance ratios of the results for both methods, where, on average, our algorithm achieved 0.02% closer to the original surface inputs. Figure 14 compares our method to Mesquite and Livesu et al. [2015] on an impeller hex-mesh, where our result has a much higher quality than the one optimized by Mesquite and achieves comparable MSJ/ASJ with Livesu et al. [2015]'s with a considerably smaller surface deviation error. We also want to emphasize that our method is fundamentally different from the two methods we compare with: [Brewer et al. 2003] and [Livesu et al. 2015] optimize the shape of hexahedra with a fixed connectivity and have the advantage of being able to deal with flipped elements, while our approach modifies the mesh connectivity to improve both the global structure and the element quality of the input mesh, but cannot recover from inverted elements.

Parameters. There are two user-controllable parameters in our current pipeline, i.e., the Hausdorff distance threshold r_h and the desired number of elements in the output mesh with respect to the input mesh $r_{|H|}$. Figure 15 shows the results of simplifying a bunny mesh with the Hausdorff distance thresholds 0.1%, 0.3%, 1%, 3% and

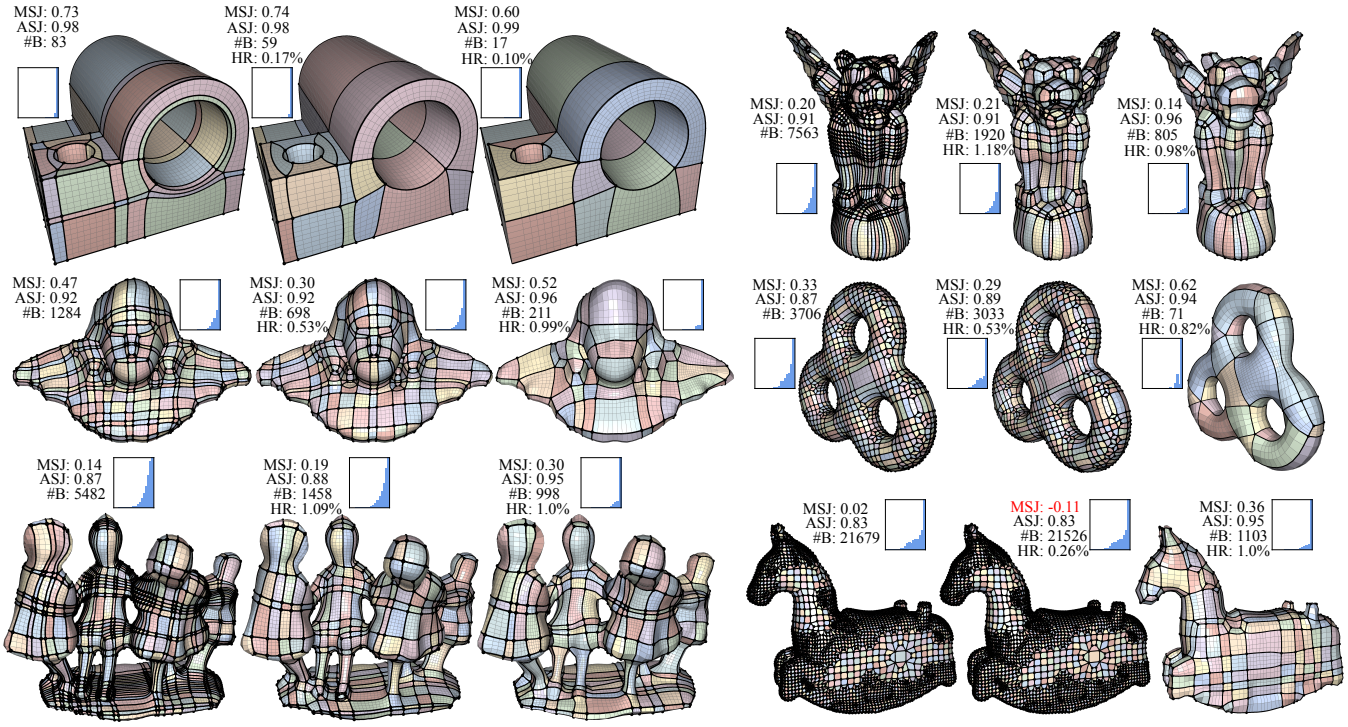


Fig. 12. Comparison with the results optimized by [Gao et al. 2015]. For each model, the input mesh is shown on the left, results from [Gao et al. 2015] is in the middle, while our result is on the right. From this comparison, we see that our method achieves much simpler structure for all models, while still preserving the surface features and the positivity of scaled Jacobians.

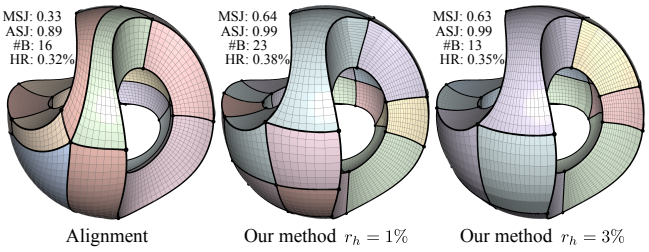


Fig. 13. Comparison with [Gao et al. 2015] for the sculpture model. By choosing a different r_h values, our method achieves better reduction of the mesh structure (right) while having better Jacobian quality than the result (left) of [Gao et al. 2015].

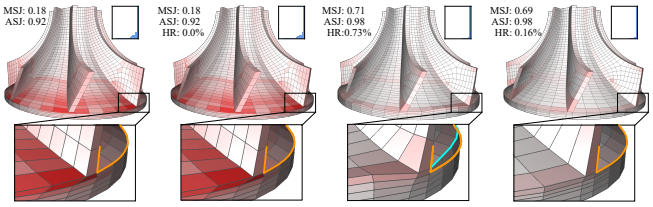


Fig. 14. Comparison with other optimization techniques. From left to right, it shows the input and the results of Mesquite [Brewer et al. 2003], the edge cone [Livesu et al. 2015] approach, and our approach, respectively. The orange lines in the magnified views are the reference line from the input mesh. The closer the corresponding segment of the boundary to this orange line, the better the surface is preserved.

10%, respectively. As expected, the larger this threshold is (i.e. allowing the loss of small surface features), the simpler the obtained structure will be. At the same time, the Jacobians of the mesh (especially minimum scaled Jacobian) may drop due to the distortion caused by the simpler structure. A small threshold might prevent the collapse of any sheets or chords, not allowing our algorithm to simplify the structure (Figure 15, top-middle). Note that, our algorithm is greedy, and the filter is applied independently to every operation –it is thus possible that a feasible solution (obtained by combining multiple operations) with a simpler and coarser structure exists, but our algorithm might fail to find it.

Given an input mesh that has $|H|$ elements, the desired element number in the output mesh is $r_{|H|} \times |H|$. In our pipeline, after each valid simplification, the element number is checked against the desired number. If the current element number is smaller than the desired number, the refinement is performed. This may affect the number of components that can be removed due to the Hausdorff distance threshold. Figure 16 shows the results of the simplification of the rockerarm mesh with different element number ratios (i.e. $r_{|H|} = 1.0$ (middle) and $r_{|H|} = 0.001$ (right), respectively). A small

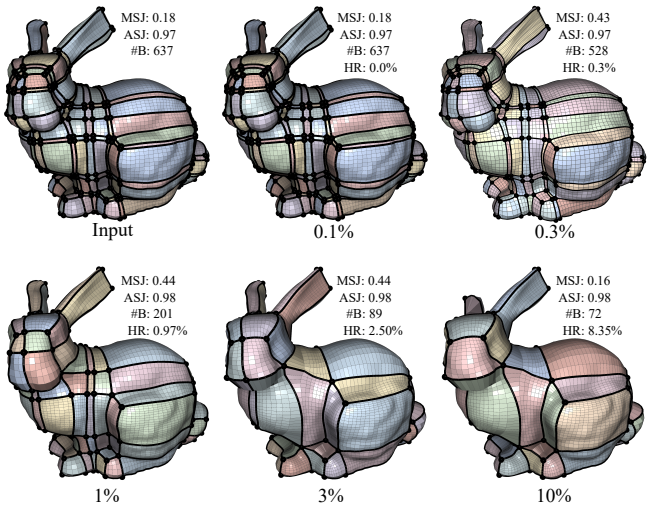


Fig. 15. The Hausdorff distance threshold is a filter applied on all simplification operations, which inhibits those changing the surface more than the user desires. A high threshold will produce a simple and coarse structure (bottom-right), while a low threshold will preserve small surface details (top-middle).

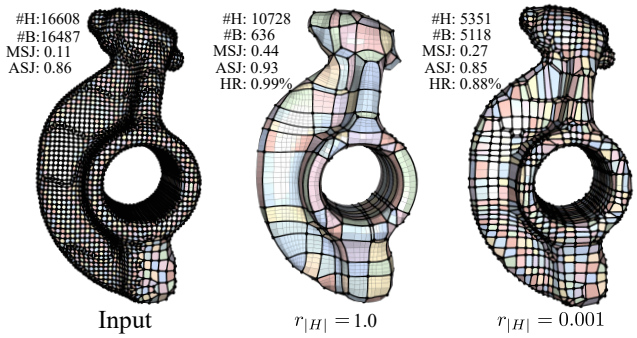


Fig. 16. Simplification results with different element number ratio with respect to the input meshes.

$r_{|H|}$ reduces the number of elements, but also limits the simplification. We thus propose to reduce the number of elements only as a postprocess (Figure 17).

Hex-mesh Coarsening. Our method can remove hexahedral elements by collapsing hexahedral sheets [Borden et al. 2002]. In particular, we first simplify the structure of a hex-mesh and then continue to remove its hexahedral elements by first tagging all the hexahedral edges as singularities to treat the individual hexahedral sheets as base complex sheets and then applying our simplification pipeline on the mesh. Figure 17 demonstrates this coarsening process on a hand model provided in [Cherchi et al. 2016]. Specifically, the second to the left image shows the mesh after the structure simplification, while the two right ones are meshes with reduced number of hexahedral elements. Note that, we achieved significant element reduction, i.e., from 32060 (input) to 30 (output).

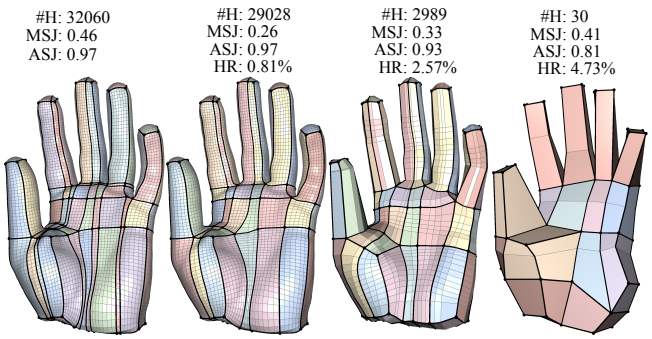


Fig. 17. Hex-mesh coarsening of a hand model. From the left to the right, it shows the input, output after structure simplification, hexahedral sheet removal after 150 iterations, and the output of hexahedral sheet removal, respectively. Note that the output mesh contains only 30 elements while still preserving the topology and to some extent the geometry of the input.

5 LIMITATIONS AND CONCLUDING REMARKS

We introduce a robust and effective algorithm to simplify the global structure of a hex-mesh. Our simplification collapses base complex sheets and chords, while redistributing the distortion using a volumetric parametrization approach. The filtering of the sheets increases the geometric fidelity, producing coarse meshes that resemble the input mesh and preserve its geometric features. We expect our contribution to have large practical impacts, since it enables to automatically create hundreds of hex-meshes when paired with automatic octree-based methods, and can be used as a coarsening post-process for all existing hex-meshing methods.

Our algorithm preserves sharp features if they are given as input — however, it is difficult to detect them automatically. Our simple dihedral angle thresholding method (which we used for all results) fails on mild features, that are then potentially lost during the simplification if they are smaller than the Hausdorff distance threshold.

Our current implementation of the solver is single-threaded, and is slow to process meshes with dense structures. Possible speed-ups can be achieved by: (1) collapsing several sheets/chords in parallel and choosing the one with the highest quality, and (2) using a parallel solver (e.g. [PARDISO 2017]) in the SLIM oprimization.

An interesting direction for future work is the extension of our method to hex-dominant meshes [Gao et al. 2017], which have been gaining popularity due to their superior flexibility and adaptivity.

ACKNOWLEDGMENTS

We thank anonymous reviewers for the insightful comments. The fertility and rocker-arm models are part of the AIM@SHAPE Shape Repository. The bunny model is courtesy of Standford graphics lab. This work was supported in part by the NSF CAREER award 1652515, NSF IIS-(1524782 and 1553329), NSFC-(61272019 and 61572292), and SZ-fund (SIRI0404201431).

REFERENCES

Y Bazilevs, L Beirão da Veiga, JA Cottrell, TJR Hughes, and G Sangalli. 2006. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences* 16, 07 (2006), 1031–1090.

Steven E. Benzley, Ernest Perry, Karl Merkley, Brett Clark, and Greg Sjaardema. 1995. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings of the 4th International Meshing Roundtable*. 179–191.

David Bommes, Timm Lempfer, and Leif Kobbelt. 2011. Global Structure Optimization of Quadrilateral Meshes. *CGF* 30, 2 (2011), 375–384.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Cláudio Silva, Marco Tarini, and Denis Zorin. 2013. Quad-Mesh Generation and Processing: A Survey. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 51–76.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph.* 28, 3, Article 77 (July 2009), 10 pages.

Michael J Borden, Steven E Benzley, and Jason F Shepherd. 2002. Hexahedral Sheet Extraction. In *Proc. of the 11th International Meshing Roundtable*. 147–152.

Xavier Bourdin, Xavier Trosseille, Philippe Petit, and Philippe Beillas. 2007. Comparison of tetrahedral and hexaedral meshes for organ finite element modeling: an application to kidney impact. In *20th International Technical Conference on the Enhanced Safety of Vehicle*. Lyon, France.

Agostino Bozzo, Daniele Panozzo, Enrico Puppo, Nico Pietroni, and Luigi Rocca. 2010. Adaptive quad mesh simplification. In *Eurographics Italian Chapter Conference 2010*.

M. Brewer, L. Diachin, P. Knupp, T. Leurent, and D. Melander. 2003. The Mesquite Mesh Quality Improvement Toolkit. In *Proc. of the 12th International Meshing Roundtable*. 239–250.

Gianmarco Cherchi, Marco Livesu, and Riccardo Scateni. 2016. Polycube Simplification for Coarse Layouts of Surfaces and Volumes. *Computer Graphics Forum* 35, 5 (2016), 11–20.

A. O. Cifuentes and A. Kalbag. 1992. A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. *Finite Elements in Analysis and Design* 12, 3–4 (1992), 313–318.

Paolo Cignoni, Cláudio Rocchini, and Roberto Scopigno. 1996. *Metro: Measuring Error on Simplified Surfaces*. Technical Report. Paris, France, France.

Joel Daniels, Cláudio T Silva, and Elaine Cohen. 2009a. Semi-regular Quadrilateral-only Remeshing from Simplified Base Domains. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 1427–1435.

Joel Daniels, Cláudio T. Silva, Jason Shepherd, and Elaine Cohen. 2008. Quadrilateral mesh simplification. *ACM Trans. Graph.* 27, 5, Article 148 (Dec. 2008), 9 pages.

Joel Daniels, II, Cláudio T. Silva, and Elaine Cohen. 2009b. Localized Quadrilateral Coarsening. In *Proceedings of the Symposium on Geometry Processing (SGP '09)*. 1437–1444.

Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-Hex Meshing using Closed-Form Induced Polycube. *Transactions on Graphics (Proc. SIGGRAPH 2016)* 35, 4 (2016).

Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient Volumetric PolyCube-Map Construction. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 97–106.

Xifeng Gao, Zhigang Deng, and Guoning Chen. 2015. Hexahedral Mesh Re-parameterization from Aligned Base-Complex. *ACM Trans. Graph. (SIGGRAPH '15)* 34, 4, Article 142 (2015), 10 pages.

Xifeng Gao, Wenzel Jakob, Marco Tarini, and Daniele Panozzo. 2017. Robust Hex-dominant Mesh Generation Using Field-guided Polyhedral Agglomeration. *ACM Trans. Graph.* 36, 4, Article 114 (July 2017), 13 pages.

Xifeng Gao, Tobias Martin, Sai Deng, Elaine Cohen, Zhigang Deng, and Guoning Chen. 2016. Structured Volume Decomposition via Generalized Sweeping. *IEEE TVCG* 22, 7 (2016), 1899–1911.

James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-Hex Mesh Generation via Volumetric PolyCube Deformation. *CGF* 30, 5 (2011), 1407–1416.

Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. L1-based Construction of Polycube Maps from Complex Shapes. *ACM Trans. Graph.* 33, 3 (2014), 25:1–25:11.

Jin Huang, Yiying Tong, Hongyu Wei, and Hujun Bao. 2011. Boundary aligned smooth 3D cross-frame field. *ACM Trans. Graph.* 30, 6, Article 143 (Dec. 2011), 8 pages.

Thomas J.R. Hughes, John A. Cottrell, and Yuri Bazilevs. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194 (2005), 4135–4195.

Yasushi Ito, Alan M. Shih, and Bharat K. Soni. 2009. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *Int. J. Numer. Meth. Engng* 77 (2009), 1809–1833.

Tengfei Jiang, Jin Huang, Yuanzhen Wang, Yiying Tong, and Hujun Bao. 2014. Frame Field Singularity Correction for Automatic Hexahedralization. *IEEE TVCG* 20, 8 (Aug. 2014), 1189–1199.

Patrick M. Knupp. 2000. Hexahedral Mesh Untangling and Algebraic Mesh Quality Metrics. In *Proceedings, 9th International Meshing Roundtable*. 173–183.

Patrick M Knupp. 2003. A method for hexahedral mesh shape optimization. *International journal for numerical methods in engineering* 58, 2 (2003), 319–332.

Leif Kobbelt, Jens Vorsatz, and Hans-Peter Seidel. 1999. Multiresolution Hierarchies on Unstructured Triangle Meshes. *Comput. Geom. Theory Appl.* 14, 1-3 (Nov. 1999), 5–24.

Nicolas Kowalski, Franck Ledoux, Matthew L. Staten, and Steve J. Owen. 2012. Fun sheet matching: towards automatic block decomposition for hexahedral meshes. *Engineering with Computers* 28, 3 (2012), 241–253.

Franck Ledoux and Jason Shepherd. 2010. Topological modifications of hexahedral meshes via sheet operations: a theoretical study. *Engineering with Computers* 26, 4 (2010), 433–447.

Benoît Leonard, Alpesh Patel, and Charles Hirsch. 2000. Multigrid acceleration in a 3D Navier-Stokes solver using unstructured hexahedral meshes with adaptation. In *Multigrid Methods VI*. Springer, 150–156.

Bo Li, Xin Li, Kexiang Wang, and Hong Qin. 2013. Surface Mesh to Volumetric Spline Conversion with Generalized Poly-cubes. *IEEE TVCG* 19, 9 (2013), 1539–1551.

Bo Li and Hong Qin. 2012. Component-aware tensor-product trivariate splines of arbitrary topology. *Computers & Graphics* 36, 5 (2012), 329–340.

Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. 2012. All-hex meshing using singularity-restricted field. *ACM Trans. Graph.* 31, 6, Article 177 (Nov. 2012), 177:1–177:11 pages.

Marco Livesu, Alessandro Muntoni, Enrico Puppo, and Riccardo Scateni. 2016. Skeleton-driven Adaptive Hexahedral Meshing of Tubular Shapes. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 237–246.

Marco Livesu, Alla Sheffer, Nicholas Vining, and Marco Tarini. 2015. Practical Hex-Mesh Optimization via Edge-Cone Rectification. *Transactions on Graphics (Proc. SIGGRAPH 2015)* 34, 4 (2015).

Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. 2013. PolyCut: monotone graph-cuts for PolyCube base-complex construction. *ACM Trans. Graph.* 32, 6 (2013), 171.

Max Lyon, David Bommes, and Leif Kobbelt. 2016. HexEx: Robust Hexahedral Mesh Extraction. *ACM Trans. Graph.* 35, 4, Article 123 (July 2016), 11 pages.

Loïc Maréchal. 2009. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *proceedings of the 18th International Meshing Roundtable*. Springer, 65–84.

MeshGems. 2015. Volume Meshing: MeshGems-Hexa. http://meshgems.com/volume-meshing-meshgems-hexa.html. (2015).

Ashish Myles, Nico Pietroni, Denis Kovacs, and Denis Zorin. 2010. Feature-aligned T-meshes. *ACM Trans. Graph.* 29, Article 117 (July 2010), 11 pages. Issue 4.

Matthias Nieser, Ulrich Reitebuch, and Konrad Polthier. 2011. CubeCover- Parameterization of 3D Volumes. *CGF* 30, 5 (2011), 1397–1406.

Steven J Owen. 1998. A Survey of Unstructured Mesh Generation Technology.. In *IMR*. 239–267.

Daniele Panozzo, Ilya Baran, Olga Diamanti, and Olga Sorkine-Hornung. 2013. Weighted Averages on Surfaces. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 32, 4 (2013), 60:1–60:12.

PARDISO. 2017. PARDISO. http://www.pardiso-project.org/. (2017).

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2, Article 16 (April 2017), 16 pages.

A. Ramos and J.A. Simões. 2006. Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur. *Medical Engineering & Physics* 28, 9 (2006), 916 – 924.

Eloi Ruiz-Gironés, Xevi Roca, and Jose Sarrate. 2014. Optimizing mesh distortion by hierarchical iteration relocation of the nodes on the CAD entities. *Procedia Engineering* 82 (2014), 101–113.

Eloi Ruiz-Gironés, Xevi Roca, Josep Sarrate, Rafael Montenegro, and José María Escobar. 2015. Simultaneous untangling and smoothing of quadrilateral and hexahedral meshes using an object-oriented framework. *Advances in Engineering Software* 80 (2015), 12–24.

National Lab Sandia. 2016. CUBIT. https://cubit.sandia.gov/. (2016).

Jason F. Shepherd and Chris R. Johnson. 2008. Hexahedral Mesh Generation Constraints. *Eng. with Comput.* 24, 3 (June 2008), 195–213.

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (July 2015), 9 pages.

C. J. Stimpson, C. D. Ernst, P. Knupp, P. P. Pébayand, and D. Thompson. 2007. The Verdict Geometric Quality Library. (2007).

Yi Su, KH Lee, and A Senthil Kumar. 2004. Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method. *Computer-Aided Design* 36, 3 (2004), 203–215.

Srinivas C. Tadepalli, Ahmet Erdemir, and Peter R. Cavanagh. 2010. A Comparison of the Performance of Hexahedral and Tetrahedral Elements in Finite Element Models of the Foot. In *ASME 2010 Summer Bioengineering Conference, Parts A and B*.

Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, and Enrico Puppo. 2010. Practical quad mesh simplification. *CGF* 29, 2 (2010), 407–418.

Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. 2011. Simple Quad Domains for Field Aligned Mesh Parametrization. *ACM Trans. Graph.* 30, 6, Article 142 (Dec. 2011), 12 pages.

Timothy J Tautges and Sarah E Knoop. 2003. Topology modification of hexahedral meshes using atomic dual-based operations. In *Proc. of the 12th International Meshing Roundtable*. 415–423.

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 545–572.

Yoshitaka Wada, Junichi Shinbori, and Masanori Kikuchi. 2006. Adaptive FEM analysis technique using multigrid method for unstructured hexahedral meshes. *Key Engineering Materials* 306 (2006), 565–570.

Thomas James Wilson, Josep Sarrate Ramos, Xavier Roca Ramón, Rafael Montenegro Armas, and José María Escobar Sánchez. 2012. Untangling and smoothing of quadrilateral and hexahedral meshes. (2012).

Hongmei Zhang, Guoqun Zhao, and Xinwu Ma. 2007. Adaptive generation of hexahedral element mesh using an improved grid-based method. *Computer-Aided Design* 39, 10 (2007), 914–928.

Yongjie Zhang and Chandrajit Bajaj. 2006. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer methods in applied mechanics and engineering* 195, 9 (2006), 942–960.

Y. J. Zhang, X. Liang, and Guoliang Xu. 2013. A robust 2-refinement algorithm in octree or rhombic dodecahedral tree based all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering* 256 (2013), 88–100.

A APPENDIX

Proposition 3.1. The number of parallel edge sets of a base complex $\mathcal{B}$ is no more than $|\mathcal{B}_E|/4$ ($|\mathcal{B}_E|$ is the number of base complex edges), so is the number of the base complex sheets in $\mathcal{B}$.

Sketch of Proof: For an edge $e_i \in E^e_{\parallel}$ and $E^e_{\parallel} \neq \emptyset$, the number of its adjacent components is larger than or equal to 1; otherwise, the base complex is not manifold. Therefore, the number of edges in $E^e_{\parallel}$ is at least 4. Since each base complex edge uniquely belongs to one parallel edge set, the number of the parallel edge sets contained in a base complex is at most $|\mathcal{B}_E|/4$. A parallel edge set uniquely identifies a base complex sheet, hence, the number of base complex sheets is at most $|\mathcal{B}_E|/4$. □

Proposition 3.2. Removing any base complex sheet from a base complex $\mathcal{B}$ monotonically reduces the number of components in $\mathcal{B}$.

Sketch of Proof: Consider the connectivity changes only. Removing a base complex sheet **S** is equivalent to discarding its middle part while collapsing the two side surfaces into one (same for the Mobiüs case). In this sense, the component set C^S is removed, and no new components are created. □

Proposition 3.3. The number of parallel patch sets of a base complex $\mathcal{B}$ is no more than $|\mathcal{B}_P|/2$ ($|\mathcal{B}_P|$ is the number of patches in $\mathcal{B}$), so is the number of the contained base complex chords in $\mathcal{B}$.

Proposition 3.4. Removing any base complex chord from a base complex $\mathcal{B}$ monotonically reduces the number of components in $\mathcal{B}$.

Sketch of Proof: Removing a base complex chord **C** is equivalent to discarding its middle part while collapsing the four side surfaces diagonally into two surfaces. Therefore, there are two diagonal collapsing directions for removing **C**. In either case, the component set C^C is removed, and no new components are introduced. □

Propositions 3.1–3.4 together guarantee that iteratively removing any base complex sheet or chord from a base complex will progressively simplify the base complex within a finite number of iterations.

Predict edge valence. A 2D analogous example is shown in Figure 18. Consider removing the blue patch p_1 diagonally, e.g., collapsing the two yellow nodes (Figure 18(a)) into one (Figure 18(b)). The yellow node in Figure 18(b) is newly created and its neighboring patch set can be exactly computed as $[\{p_1, p_5\} - \{p_1\}] \cup [\{p_1, p_2, p_3\} - \{p_1\}] = \{p_2, p_3, p_5\}$. The valences of edges that are affected by the collapsing can be similarly computed.

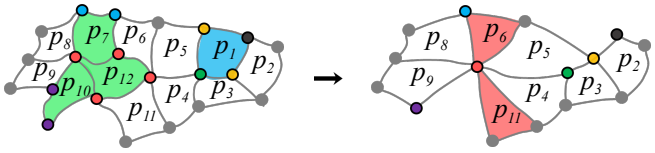


Fig. 18. Given a 2D structure (left), removing its colored patches leads to valence changes of some of its nodes (right), where nodes (left) with the same color belong to the same collapsing group. The valences of all the nodes (right) are exactly the same as the results computed by Equation 6.

For edges that are connected to those to-be-collapsed edges, they can be considered as newly created edges that are collapsed to themselves. Thus, Equation 6 can be used to update their valences. For instance, the green node of the blue patch p_1 in Figure 18(a) is collapsed to itself after collapsing p_1. Its valence can then be updated using Equation (6) as $[\{p_1, p_3, p_4, p_5\} - \{p_1\}] \cup [\{p_1, p_3, p_4, p_5\} - \{p_1\}] = \{p_3, p_4, p_5\}$. In fact, Equation 6 can be generalized to compute the valences of all low-dimensional elements (e.g., vertices, edges, and faces) of a 2D and 3D mesh after simplification as long as the collapsing groups are known. Here, all the elements in a collapsing group will be replaced by a new element in the simplified structure. Figure 18(a) illustrates the collapsing of groups of nodes in 2D (e.g., the nodes of the green patches). In this case, all blue nodes, all red nodes, and all purple nodes collapse into one node, respectively. Furthermore, Equation 6 does not require the resulting simplified mesh to be all quad or all hex. For example, after the collapsing of those groups of nodes in Figure 18(a), two triangle patches are resulted (i.e., the red patches in Figure 18(b)). The valences of those involved edges can still be accurately computed using Equation (6).