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

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 and it demonstrates 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:

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).

Plycube 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 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 Shephard 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.
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.

Singularity Alignment.

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 $V$ into topologically cube-like components, which provides a coarse structural tessellation of $V$. 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 edges. 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.
where $E_b(V)$ 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 $\lambda_c = 10^5$ 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 $D_b(V)$ of each hexahedron as the sum of the distortions of its eight tetrahedra $D_t(V)$.

$$E_D(V) = \sum_{h \in H} D_b(V) = \sum_{h \in H} \sum_{t \in h} M_t D_t(V)$$

$$D_t(V) = ||J_t||_F^2 + ||J_t^{-1}||_F^2$$

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 right-angled tetrahedron, we 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
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:

\[
E_B(V) = \sum_{v \in C} ||v - \tilde{v}||^2 + \sum_{v \in L} ||v - \tilde{v} - a_l \tilde{d}_l||^2
\]

\[
+ \sum_{v \in S} ||\tilde{n}(v - \tilde{v})||^2
\]

where \( \tilde{v} \) is the closest surface position for each boundary vertex, \( \tilde{d}_l \) is the feature line tangent at \( \tilde{v} \), \( a_l \) is an auxiliary variable added to the system for feature line constraints, and \( \tilde{n} \) 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

\[
\min V E(V) = E_D(V) + \lambda B E_B(V),
\]

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 \( \beta \)-ring neighborhood of the to-be-collapsed sheet/chord.
185:6 • Xifeng Gao, Daniele Panozzo, Wenping Wang, Zhigang Deng, and Guoning Chen

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 $\beta$ 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 $\beta$ 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 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 $\tau_e$ 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

$$C'(e') = (C(e_0) - C(r_0)) \cup (C(e_1) - C(r_1)) \cup \ldots \cup (C(e_n) - C(r_n))$$

where $C(e)$ is the set of the neighboring components of $e$ in the base complex before simplification, $C(r)$ 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 rank- ing 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.](image)

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 \( \#B_{\text{out}} - \#B_{\text{in}} \)/\( \#B_{\text{in}} \). HR is defined as \( \max[|d(S_i, S_o)|, |d(S_o, S_i)|]/D_{\text{box}}(S_i \cup S_o) \), 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.
$D_{box}(X)$ 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.

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 J1-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 $r_k = 1\%$ reduces it to 23 (Figure 13 (middle)). However, by setting $r_k = 3\%$ 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.

<table>
<thead>
<tr>
<th>Models</th>
<th>#H</th>
<th>#B</th>
<th>MSJ/ASJ/Std.</th>
<th>Output #H</th>
<th>#B</th>
<th>MSJ/ASJ/Std.</th>
<th>HR (%)</th>
<th>R</th>
<th>Time (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane1 (Figure 9)</td>
<td>4972</td>
<td>3661</td>
<td>0.03/0.84/2.45</td>
<td>4685</td>
<td>200</td>
<td>0.45/0.93/0.84</td>
<td>0.89</td>
<td>95%</td>
<td>10.2</td>
</tr>
<tr>
<td>airplane2 (Figure 8)</td>
<td>6118</td>
<td>4319</td>
<td>0.02/0.85/2.49</td>
<td>6154</td>
<td>62</td>
<td>0.55/0.94/0.65</td>
<td>0.80</td>
<td>99%</td>
<td>11.0</td>
</tr>
<tr>
<td>chair1 (Figure 9)</td>
<td>11866</td>
<td>9322</td>
<td>0.03/0.84/2.03</td>
<td>10314</td>
<td>560</td>
<td>0.29/0.91/0.84</td>
<td>0.97</td>
<td>94%</td>
<td>46.1</td>
</tr>
<tr>
<td>cup (Figure 9)</td>
<td>48341</td>
<td>48075</td>
<td>0.03/0.87/1.96</td>
<td>40144</td>
<td>382</td>
<td>0.45/0.96/0.41</td>
<td>0.90</td>
<td>99%</td>
<td>227.0</td>
</tr>
<tr>
<td>deckel (Figure 9)</td>
<td>53658</td>
<td>53116</td>
<td>0.03/0.84/3.50</td>
<td>48822</td>
<td>2122</td>
<td>0.24/0.94/0.96</td>
<td>1.0</td>
<td>96%</td>
<td>504.1</td>
</tr>
<tr>
<td>elk (Figure 1)</td>
<td>24916</td>
<td>23609</td>
<td>0.01/0.82/3.49</td>
<td>15156</td>
<td>324</td>
<td>0.43/0.95/0.58</td>
<td>0.96</td>
<td>98%</td>
<td>87.0</td>
</tr>
<tr>
<td>kiss (Figure 9)</td>
<td>18418</td>
<td>18360</td>
<td>0.03/0.84/2.27</td>
<td>15492</td>
<td>1174</td>
<td>0.44/0.94/0.68</td>
<td>1.0</td>
<td>94%</td>
<td>47.6</td>
</tr>
<tr>
<td>toy1 (Figure 9)</td>
<td>18947</td>
<td>18883</td>
<td>0.12/0.83/2.60</td>
<td>12972</td>
<td>481</td>
<td>0.42/0.94/0.61</td>
<td>0.89</td>
<td>97%</td>
<td>32.9</td>
</tr>
<tr>
<td>fertility (Figure 10)</td>
<td>13584</td>
<td>1352</td>
<td>0.35/0.91/0.77</td>
<td>1060</td>
<td>156</td>
<td>0.12/0.92/2.24</td>
<td>0.95</td>
<td>88%</td>
<td>11.9</td>
</tr>
<tr>
<td>impeller (Figure 10)</td>
<td>11174</td>
<td>878</td>
<td>0.18/0.92/1.07</td>
<td>11496</td>
<td>152</td>
<td>0.73/0.99/0.11</td>
<td>0.29</td>
<td>83%</td>
<td>5.7</td>
</tr>
<tr>
<td>armadillo (Figure 11)</td>
<td>78376</td>
<td>7906</td>
<td>0.27/0.93/0.64</td>
<td>57567</td>
<td>549</td>
<td>0.29/0.97/0.32</td>
<td>0.99</td>
<td>91%</td>
<td>293.3</td>
</tr>
<tr>
<td>bunny (Figure 11)</td>
<td>81637</td>
<td>1324</td>
<td>0.14/0.93/0.71</td>
<td>71440</td>
<td>153</td>
<td>0.43/0.98/0.21</td>
<td>0.93</td>
<td>88%</td>
<td>110.7</td>
</tr>
<tr>
<td>carter (Figure 11)</td>
<td>64911</td>
<td>1101</td>
<td>0.22/0.94/0.78</td>
<td>51100</td>
<td>141</td>
<td>0.28/0.97/0.38</td>
<td>0.98</td>
<td>87%</td>
<td>38.7</td>
</tr>
<tr>
<td>dragon (Figure 11)</td>
<td>111478</td>
<td>12488</td>
<td>0.16/0.92/0.74</td>
<td>89348</td>
<td>1927</td>
<td>0.28/0.96/0.33</td>
<td>1.0</td>
<td>85%</td>
<td>348.2</td>
</tr>
<tr>
<td>pegasusus (Figure 1)</td>
<td>120344</td>
<td>9745</td>
<td>0.23/0.94/0.55</td>
<td>94492</td>
<td>1573</td>
<td>0.33/0.97/0.34</td>
<td>1.0</td>
<td>84%</td>
<td>316.4</td>
</tr>
</tbody>
</table>

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 distribution of scaled Jacobians (in percentage) of all elements in a mesh. HR indicates the Hausdorff distance, and R is the percentage of reduction.

<table>
<thead>
<tr>
<th>Models</th>
<th>#H</th>
<th>#B</th>
<th>MSJ/ASJ/Std.</th>
<th>[Gao et al. 2015]</th>
<th>#H</th>
<th>#B</th>
<th>MSJ/ASJ/Std.</th>
<th>HR (%)</th>
<th>R</th>
<th>Time (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3torus</td>
<td>4664</td>
<td>3706</td>
<td>0.33/0.87/1.47</td>
<td>42008</td>
<td>71</td>
<td>0.02/0.94/0.47</td>
<td>0.82</td>
<td>98%</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>isodore</td>
<td>21695</td>
<td>21679</td>
<td>0.02/0.83/2.73</td>
<td>13744</td>
<td>1103</td>
<td>0.26/0.95/0.66</td>
<td>1.0</td>
<td>95%</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>joint</td>
<td>17784</td>
<td>83</td>
<td>0.73/0.98/0.07</td>
<td>19422</td>
<td>17</td>
<td>0.66/0.99/0.05</td>
<td>0.10</td>
<td>80%</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>angel_1</td>
<td>14068</td>
<td>1284</td>
<td>0.47/0.92/0.61</td>
<td>11891</td>
<td>211</td>
<td>0.52/0.96/0.40</td>
<td>0.99</td>
<td>84%</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>dancing children</td>
<td>35293</td>
<td>5482</td>
<td>0.14/0.87/1.13</td>
<td>30123</td>
<td>998</td>
<td>0.30/0.95/0.50</td>
<td>1.0</td>
<td>82%</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>gargoyl</td>
<td>25669</td>
<td>7553</td>
<td>0.20/0.91/0.82</td>
<td>22524</td>
<td>805</td>
<td>0.14/0.96/0.46</td>
<td>0.98</td>
<td>89%</td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>

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 any- more, 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 \( r_{|M|} \). 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.

Alignment Our method
Our method

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

Given an input mesh that has $|H|$ elements, the desired element number in the output mesh is $r_H |H|$. In our pipeline, after each 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. $r_H |H| = 0.001$ (right), respectively). A small
r 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).

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 optimization.

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 Stanford 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 (SIRI040201431).

REFERENCES


Wiley Online Library, 1399–1411.


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 $e_1 \in E^e_\mathcal{B}$ and $E^e_\mathcal{B} \neq \emptyset$, 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 $E^e_\mathcal{B}$ 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 $|\mathcal{B}_E|/4$. \(\square\)

**Proposition 3.3** 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 Mobius case). In this sense, the component set $C^S$ is removed, and no new components are created. \(\square\)

**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. \(\square\)

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 $\{p_1\} = \{p_2, p_3, p_5\}$. The valences of edges that are affected by the collapsing can be similarly computed.

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).