All-Hex Meshing using Singularity-Restricted Field

Yufei Li[†] Yang Liu[*] Weiwei Xu[*] Wenping Wang[†] Baining Guo[*]
[*]Microsoft Research Asia [†]The University of Hong Kong

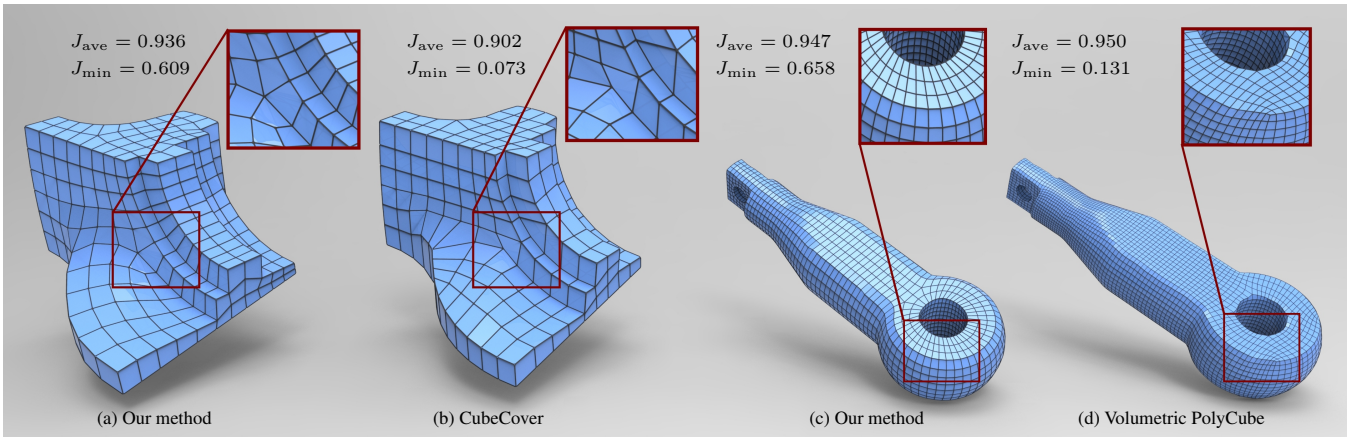


Figure 1: *High quality all-hex meshes generated by our method. Comparisons with CubeCover [Nieser et al. 2011] and volumetric PolyCube [Gregson et al. 2011] demonstrate that the hex meshes by our method are superior in mesh quality (the minimal scaled Jacobian of hexes is shown in the figure, bigger is better) and singularity placement (see the zoom-in views for comparison).*

Abstract

Decomposing a volume into high-quality hexahedral cells is a challenging task in geometric modeling and computational geometry. Inspired by the use of cross field in quad meshing and the CubeCover approach in hex meshing, we present a complete all-hex meshing framework based on *singularity-restricted field* that is essential to induce a valid all-hex structure. Given a volume represented by a tetrahedral mesh, we first compute a boundary-aligned 3D frame field inside it, then convert the frame field to be singularity-restricted by our effective topological operations. In our all-hex meshing framework, we apply the CubeCover method to achieve the volume parametrization. For reducing degenerate elements appearing in the volume parametrization, we also propose novel tetrahedral split operations to preprocess singularity-restricted frame fields. Experimental results show that our algorithm generates high-quality all-hex meshes from a variety of 3D volumes robustly and efficiently.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: all-hex meshing, singularity-restricted field, 3D frame field.

Links: DL PDF

1 Introduction

All-hex meshes possess nice numerical properties, such as a reduced number of elements and high approximation accuracy in physical simulation and mechanical engineering [Shimada 2006; Shepherd and Johnson 2008], but it is still a challenging task to automatically decompose a general 3D volume with complex boundary into high-quality hex elements [Owen 1998]. It is generally accepted that a high-quality all-hex mesh should have three properties: (1) Boundary conforming. The generated all-hex mesh should conform to the boundary surface of the 3D volume. (2) Feature alignment. Feature edges in the input 3D volume should be aligned. (3) Most important, low distortion. A single inverted hex can ruin finite element simulation. To achieve this point, singularities, the interior edges with other than four incident hexes in an all-hex mesh, should be introduced and embedded in the interior of the volume to reduce distortion.

Recently several methods have been proposed for the generation of high-quality all-hex meshes. Octree-based method [Maréchal 2009] and volumetric PolyCube [Gregson et al. 2011] embed axis-aligned boxes into the 3D volume to produce all-hex meshes. However, since these two methods do not generate interior singularities, large distortion might be introduced in the resulting all-hex meshes. Another approach, CubeCover [Nieser et al. 2011], generates all-hex meshes with the guidance of a *valid* 3D frame field via global parametrization. It first constructs a 3D frame field with the assistance of manually designed *meta-mesh*, computes a volumetric parametrization guided by this field, and then extracts an all-hex mesh accordingly. However, for a 3D volume with complex boundary, it requires a lot of manual efforts for a skilled user to design a suitable meta-mesh that can characterize the complex geometry. How to generate a valid frame field automatically is still an open problem. Recently Huang *et al.* [2011] design boundary-aligned 3D frame fields for guiding hex meshing but generate only hex-dominant meshes, not all-hex meshes.

In this paper we present a method for generating a high-quality all-hex mesh from a given tetrahedral mesh. A key ingredient of our work is a novel frame field called *singularity-restricted field*

(SRF). It is known that 24 types of singularities can be found in a frame field that locally describe the orientational change of the frame field. However, only ten types of these 24 can exist in a valid all-hex structure [Nieser et al. 2011]. We call a frame field an SRF if its singularities belong to these ten types. A basic advantage of the SRF is that it is essential for all-hex meshing because only an SRF can admit a valid all-hex structure. Another advantage of the SRF is that it can be constructed under relatively weak conditions. We present a technique for converting an arbitrary frame field to an SRF under reasonable assumptions (see Section 3.3) about the input frame field and the underlying tetrahedral mesh. Finally, the SRF has an important practical advantage: with careful volume parametrization, a smooth SRF almost always leads to successful all-hex meshing in practice despite the fact that the SRF is not theoretically sufficient for all-hex meshing. We have verified this practical advantage through extensive testing.

Another crucial component of our algorithm is the SRF-guided volume parametrization, which can effectively deal with degenerate and flipped elements in the parametrization domain. These elements lead to missing and erroneous iso-lines and thus make extracting an all-hex mesh extremely difficult. Our technique combines a degenerate-element preprocessing step for volume parametrization and an edge collapse operation on the iso-curve network. The degenerate-element preprocessing step detects and fixes most degenerate elements before parametrization by eliminating unsuitable combinations of singular edges. We resolve most issues raised by flipped elements by collapsing erroneous edges from the iso-curve network whose vertices share the same parametrization coordinate.

We have developed a system for all-hex meshing based on our algorithm and found the system works well in practice based on our testing of a wide variety of natural objects and CAD models. For all the objects and models tested, our algorithm generates all-hex meshes successfully and the resulting meshes are of higher quality than those produced by existing methods. The comparisons in Figure 1 show that the shapes of hex elements in our results are more cube-like than those produced by CubeCover [Nieser et al. 2011] and volumetric PolyCube [Gregson et al. 2011]. In particular, our results (Figure 1-a&c) do not exhibit the surface singularities or distortion possessed by the results in Figure 1-b&d. Quantitatively our results are also significantly better as measured by the mesh quality $J_{\min}$ (the minimum of the scaled Jacobian).

2 Related Work

All-hex meshing has been studied for several decades. Thorough surveys are available in [Owen 1998; Shimada 2006; Shepherd and Johnson 2008]. Most methods used in industry, such as multiple sweeping [Shepherd et al. 2000], paving and plastering [Staten et al. 2005] are still semi-automatic and require the user to decompose the model for suitable mappings. Sheffer *et al.* [1999] develop an automatic method to decompose the volume by Voronoi graph but the computation is very sensitive to the boundary. Carbonera and Shepherd [2006] give a constructive approach to constrained hex meshing by using Geode-Template. In general, generating a high-quality and boundary-conforming hex mesh for a complex model takes hours or days even for a skilled user since the partition and sizing is crucial to the mesh quality. Due to the simplicity and robustness, the grid-based methods are still dominating in the meshing community. Maréchal [2009] presents a novel Octree-based method to reduce the distortion of hexes and preserve sharp edges. Another approach is based on PolyCube which maps the volume to several jointed boxes [Tarini et al. 2004; Han et al. 2010]. Recently Gregson *et al.* [2011] present a volumetric PolyCube method to convert a volume to an all-hex mesh and further reduce the overall distortion greatly. However these methods inherit the intrinsic disadvantages of grid-based methods, i.e. the choices of box orientations and placing all singularities on the boundary surface, which may still introduce large distortion and mis-matched features.

Quadrilateral meshing has gained great success by using surface parametrization techniques [Tong et al. 2006; Kälberer et al. 2007; Bommes et al. 2009]. Both Kalberer *et al.* [2007] and Bommes *et al.* [2009] present theoretical and practical approaches to find a proper surface parametrization based on a cross field defined on the triangular surface. It is natural to consider how to extend the quad meshing technique to hex meshing. Nieser *et al.* [2011] generalize QuadCover [Kälberer et al. 2007] to 3D. The Morse-Smale complex technique [Dong et al. 2006; Huang et al. 2008] is another approach for quad meshing, but to the best of our knowledge there is no clear theoretical result on its 3D generalization to all-hex meshing.

Frame field is N-coupled vectors defined at every point in the N-dimensional space. It is also called a *cross field* if the N-coupled vectors form an orthonormal frame. The cross field in surface has been studied thoroughly in [Palacios and Zhang 2007; Ray et al. 2008; Ray et al. 2009; Crane et al. 2010] for surface meshing and texture mapping. However, there is little work on 3D frame field compared to the surface case. Nieser *et al.* [2011] design the frame field using meta-mesh, whose creation is manual and hard to design for complex volumes. Huang *et al.* [2011] construct a boundary-aligned frame field by modeling its smoothness with a convenient spherical-harmonic representation.

3 Singularity-Restricted Field Computation

In this section, we first introduce the singularity of a 3D frame field as described in [Nieser et al. 2011] and define the singularity-restricted field in Section 3.1. Since it is difficult to directly compute an SRF, we present an efficient method to compute a boundary-aligned 3D frame field in Section 3.2 and apply topological operations to convert a frame field to an SRF in Section 3.3.

3.1 Singularity-Restricted Field

A 3D frame $\mathbf{F}$ is defined by an ordered tuple of three vectors $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$. Frame $\mathbf{F}$ is called *right-handed* if $\mathbf{U} \times \mathbf{V} \cdot \mathbf{W} > 0$. Two right-handed frames $\mathbf{F}_i$ and $\mathbf{F}_j$ are casted to an equivalent class if $\mathbf{F}_i$ can be permuted to $\mathbf{F}_j$, or reversely. It is known that there are 24 equivalent classes since the permutations form the chiral cubical symmetry group $\mathcal{G}$ (any map in $SO(3)$ which maps coordinate axes to coordinate axes).

Given an input tetrahedral mesh $\mathcal{V}$ bounded by a closed triangular surface $\mathcal{S}$, right-handed frames defined on each tet form a discrete 3D frame field. For two adjacent tets $\mathbf{s}$ and $\mathbf{t}$ with one common face, we characterize the closeness of the associate frames $\mathbf{F_s}$ and $\mathbf{F_t}$ by the best permutation defined as follows:

$$\Pi_{\mathbf{st}} := \arg\min_{\mathbf{P}\in\mathcal{G}} \|[\mathbf{U_s}|\mathbf{V_s}|\mathbf{W_s}] - [\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]\mathbf{P}^T\|_F, \quad (1)$$

where $\|\cdot\|_F$ is the Frobenius matrix norm. We call $\Pi_{\mathbf{st}}$ the matching matrix of $\mathbf{F_s}$ and $\mathbf{F_t}$.

The singularity of a 3D frame field is naturally defined on the edge of the tet mesh. For an interior edge $\mathbf{e}$, the concatenation of matchings between all adjacent tets $(\mathbf{t}_0, \ldots, \mathbf{t}_k, \mathbf{t}_0)$ defines the type of $\mathbf{e}$:

$$\text{type}(\mathbf{e}, \mathbf{t}_0) = \Pi_{\mathbf{t}_k\mathbf{t}_0} \circ \Pi_{\mathbf{t}_{k-1}\mathbf{t}_k} \circ \cdots \circ \Pi_{\mathbf{t}_1\mathbf{t}_2} \circ \Pi_{\mathbf{t}_0\mathbf{t}_1}. \quad (2)$$

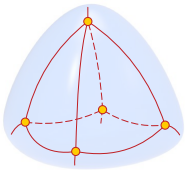
$\mathbf{e}$ is called a *singular edge* if its type is not the identity matrix.

We call the curve network formed by the singular edges a *singular graph* and the vertices on the singular graph *singular vertices*. The singular graph of a 3D frame field has a nice property:

Proposition 1 *Any singular edge of a singular graph does not end inside the volume.*

Proof: see Appendix A. ■

We call a 3D frame field *singularity-restricted* if its type of edges fall into the set $\mathcal{R} := \{I, R_u^k, R_v^k, R_w^k \mid k \in \{1,2,3\}\}$, where R_u, R_v, R_w represent the 90 degree rotations around $u-, v-, w-$ coordinate axes, respectively. These restricted edge types are demanded by valid all-hex structures [Nieser et al. 2011]. We call edges with types in $\mathcal{R}$ *proper*, in $\mathcal{G}/\mathcal{R}$ *improper*.



Remark: The singularity-restricted property is essential but not sufficient to induce an all-hex structure whose irregular edges comply with the singular graph of the field. The existence of all-hex structures is affected by the global combinatoric topology of the singular graph. For instance, in the right inset, the singular graph prohibits the existence of all-hex meshes due to triangle loops. Finding the sufficient condition of the existence of all-hexes for an arbitrary SRF is beyond our scope.

3.2 Boundary-aligned 3D Frame Field Generation

Automatic generation of 3D frame fields is non-trivial since it is difficult to measure the smoothness of a 3D frame field. Unlike the cross field on surface [Ray et al. 2008; Bommes et al. 2009; Crane et al. 2010], the smoothness of a 3D frame field cannot be encoded by only one parameter. Mathematically, the smoothness of two frames $\mathbf{F_s}, \mathbf{F_t}$ should reflect the closeness from the transformation between $\mathbf{F_s}$ and $\mathbf{F_t}$ to the chiral cubical symmetry group $\mathcal{G}$. We extend the permutation approach in [Liu et al. 2011] to measure the smoothness of a 3D frame field.

Smoothness. Two adjacent frames $\mathbf{F_s}$ and $\mathbf{F_t}$ are deemed to be perfectly smooth, if they can be permuted to each other, i.e.,

$$[\mathbf{U_s}|\mathbf{V_s}|\mathbf{W_s}] = [\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]\mathbf{P_{st}},$$

where $\mathbf{P_{st}} \in \mathcal{G}$ represents a column permutation. For two arbitrary frames $\mathbf{F_s}$ and $\mathbf{F_t}$, we approximate the column permutation by $\widetilde{\mathbf{P}}_{\mathbf{st}} = [\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]^{-1}[\mathbf{U_s}|\mathbf{V_s}|\mathbf{W_s}]$. We define the closeness from $\widetilde{\mathbf{P}}_{\mathbf{st}}$ to $\mathcal{G}$ as

$$\begin{aligned} E_{\mathbf{st}} = &\sum_i \left[H(\widetilde{\mathbf{P}}_{\mathbf{st}}[\cdot,i]) + H(\widetilde{\mathbf{P}}_{\mathbf{st}}[i,\cdot])\right] \\ &+ \sum_i \left[(\widetilde{\mathbf{P}}_{\mathbf{st}}[i,\cdot]^2 - 1)^2 + (\widetilde{\mathbf{P}}_{\mathbf{st}}[\cdot,i]^2 - 1)^2\right] \\ &+ (\det(\widetilde{\mathbf{P}}_{\mathbf{st}}) - 1)^2 \end{aligned}$$

where $\widetilde{\mathbf{P}}_{\mathbf{st}}[i,\cdot]$ and $\widetilde{\mathbf{P}}_{\mathbf{st}}[\cdot,i]$ denote the i-th row and column vector respectively, and $H(\eta) = \eta_x^2\eta_y^2 + \eta_y^2\eta_z^2 + \eta_z^2\eta_x^2$. The first term is used to constrain that there is only one nonzero component in each row or column of $\mathbf{P_{st}}$, the second term enforces the row/column vectors be unit vectors, the third term imposes the constraint that $\mathbf{F_s}$ and $\mathbf{F_t}$ keep the same orientation. If we restrict the frame field to be orthonormal, $E_{\mathbf{st}}$ can be simplified by removing its second and third term, and the inverse of matrix is avoided due to $[\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]^{-1} = [\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]^T$.

Compared with the smoothness measure proposed in [Huang et al. 2011] where they integrate the squared difference $(H([\mathbf{U_s}|\mathbf{V_s}|\mathbf{W_s}]^T\eta) - H([\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]^T\eta))^2$ over the unit sphere S^2, our formulation performs the measurement on the transformation directly without any integration. Actually it is not difficult to derive that our measurement on the orthonormal frame field is exactly the same as Huang *et al.*'s integration up to a scale of $\frac{16\pi}{315}$.

Proposition 2 *If frames $\mathbf{F}_s$ and $\mathbf{F}_t$ are orthonormal frames, we have*

$$\begin{aligned} &\int_{\mathbb{S}^2} (H([\mathbf{U_s}|\mathbf{V_s}|\mathbf{W_s}]^T\eta) - H([\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]^T\eta))^2 \,\mathrm{d}\eta \\ &= \frac{16\pi}{315}\sum_{i=1}^{3}\left[H(\widetilde{\mathbf{P}}_{\mathbf{st}}[\cdot,i]) + H(\widetilde{\mathbf{P}}_{\mathbf{st}}[i,\cdot])\right] \end{aligned}$$

Proof: It is known that $H(R^T\eta) = H(\eta)$ for any rotation matrix R. The integration on the left side can be rewritten as $\int_{\mathbb{S}^2}(H(\widetilde{\mathbf{P}}_{\mathbf{st}}\eta) - H(\eta))^2\,\mathrm{d}\eta$. We represent $\widetilde{\mathbf{P}}_{\mathbf{st}}$ by $R_X(\alpha)R_Y(\beta)R_Z(\gamma)$ in XYZ Euler angles. Thus the equality can be easily verified by symbolic computation. We provide the expanded form of the right side of the equality in Appendix B. ■

Frame field initialization. For the purpose of boundary-conforming hex meshing, one direction of the boundary frames should be the surface normal. Thus we design the boundary tet frames by surface normals and a smooth surface cross field, for instance, a principal-direction dominant frame field. We then propagate the boundary tet frames into interior tets. For any interior tet, its frame is assigned to be the same as the one of its nearest boundary tet.

Frame field optimization. The initial frame field is usually non-smooth around the medial axis of the volume due to our simple frame propagation scheme. We further smooth the frame field by minimizing the following energy function:

$$E := \sum_{\mathbf{e}} \sum_{\mathbf{s},\mathbf{t}\in\mathcal{N}(\mathbf{e})} E_{\mathbf{st}}. \tag{3}$$

Here $\mathcal{N}(\mathbf{e})$ denotes the one ring tets around the tet edge $\mathbf{e}$. Any two frames in $\mathcal{N}(\mathbf{e})$ are picked to compute their smoothness. We notice in the experiments that this choice makes the frame field smoother than only minimizing the summation of $E_{\mathbf{st}}$ between adjacent tets. We represent the orthonormal frame as follows:

- Any interior frame is represented by the column vectors of $R_x(\alpha)R_y(\theta)R_z(\gamma)$, where α, θ, γ are Euler angles.
- Any boundary frame is represented by $\{\cos\theta \cdot T_1 + \sin\theta \cdot T_2, -\sin\theta \cdot T_1 + \cos\theta \cdot T_2, N\}$. T_1, T_2 are orthonormal bases of boundary triangular face and N is the face normal. This representation helps to fix the normal direction in the optimization.

For a non-orthonormal frame field, we represent each frame using three pairs of spherical angles $(\alpha_1,\beta_1), (\alpha_2,\beta_2), (\alpha_3,\beta_3)$. α_3 and β_3 of a boundary frame are fixed during the optimization.

We minimize the nonlinear function E by the efficient L-BFGS method [Liu and Nocedal 1989] which only requires the function value and its gradient. The gradient of E can be computed easily via matrix algebra $\partial(M^{-1}) = -M^{-1}(\partial M)M^{-1}, \partial(\det(M)) = \det(M)\,\mathrm{Trace}(M^{-1}\partial M)$ and the chain rule. In experiments, the convergence criterion is set as $\frac{\|E_{i+1}-E_i\|}{\|E_i\|} < 10^{-4}$ and E converges in less than 300 iterations. Figure 2 demonstrates three optimized frame fields on an ellipsoidal volume.

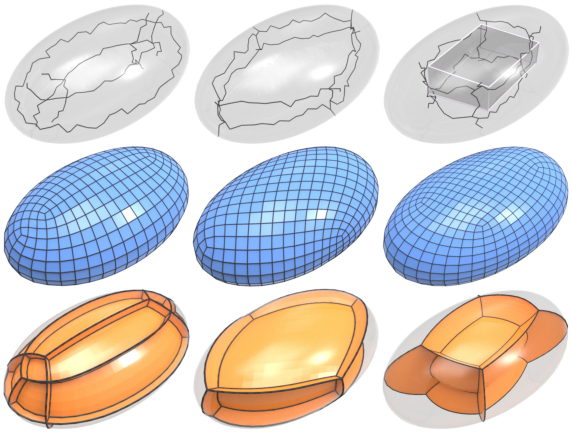


Figure 2: *Three kinds of 3D frame fields on an ellipsoidal volume. The singular graphs are in black on the first row. The second and third rows show the hex meshes and singular structures (interior iso-surfaces passing through singular edges) respectively.*

Remark. Although the smoothness measure of two adjacent orthnormal frames in [Huang et al. 2011] is the same to ours, their formulation of the objective function of field optimization is not the direct summation of $E_{\mathbf{st}}$ as defined in Eqn 3. In Section 6, we shall show the advantage of our formulation that can generate smoother frame fields.

3.3 Singularity-Restricted Field Conversion

Since there is no global control like the meta-mesh used in [Nieser et al. 2011], the edge types of a frame field do not always fall into the restricted types $\mathcal{R} := \{I, R_u^k, R_v^k, R_w^k \mid k \in \{1,2,3\}\}$. In this section, we propose two operations, *matching adjustment* and *improper singular edge collapse*, to convert a 3D frame field to a singularity-restricted field.

3.3.1 Matching adjustment

Since the initial matching $\Pi_{\mathbf{st}}$ is computed by the local information only (see Eqn. 1), it might introduce improper singular edges. Hence, for any triangular face (shared by $\mathbf{s}, \mathbf{t}$) which contains improper singular edges, we perform the following steps to greedily adjust its matching $\Pi_{\mathbf{st}}$ until no matching can be changed:

1. Sort all 24 possible permutation matrices $\{\mathbf{P}_i\}$ corresponding to $\mathcal{G}$ by ascending order according to the value $\|[\mathbf{U_s}|\mathbf{V_s}|\mathbf{W_s}] - [\mathbf{U_t}|\mathbf{V_t}|\mathbf{W_t}]\mathbf{P}_i^T\|_2$;
2. Tentatively replace $\Pi_{\mathbf{st}}$ by each sorted $\mathbf{P}_i$ and recompute the types for the three triangular edges. If all of them fall into $\mathcal{R}$ with at least one being the identity, $\mathbf{P}_i$ is accepted as the new matching matrix.
3. if all $\mathbf{P}_i$ fail the singular type criterion in Step 2, restore the original matching matrix $\Pi_{\mathbf{st}}$.

Notice that triangular face $\triangle_{pqr}$ may contain two singular edges $\overrightarrow{pq}, \overrightarrow{qr}$ with the same type $T_* \in \mathcal{R}$, we can reduce the number of proper singular edges by reusing the matching adjustment algorithm with a simple modification: in step 2, we replace $\Pi_{\mathbf{st}}$ by each sorted $\mathbf{p}_i$ until $\triangle_{pqr}$ has only one proper singular edge.

This greedy operation can eliminate most improper singular edges without changing the topology of the tet mesh, leaving the rest improper singular ones to the following *improper singular edge collapse* operation.

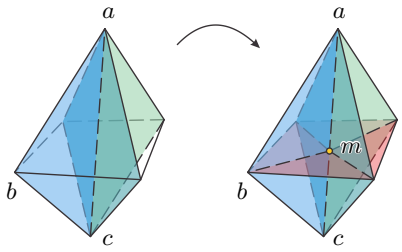
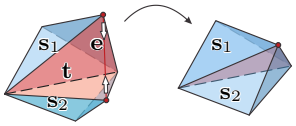


Figure 3: *Edge split operation. By splitting edge ac at its middle point m, every tet around ac is divided to two small tets. Let the frames of small tets inherit the original frames of its parent tets, the matching matrices associated to the split faces containing a or c are not changed and the matching matrices associated to other split faces are identity. It is easy to derive that am and mc keep the same edge types as ac and other new edges connecting m are nonsingular by Eqn. 2. For instance bm is nonsingular and am has the same edge type as ac before splitting.*

3.3.2 Improper singular edge collapse



Edge collapse is the most effective operation used in tetrahedral mesh simplification [Trotts et al. 1998]. It is intuitive to resolve the improper singular edge issue by collapsing them. The right inset shows the collapse operation on edge $\mathbf{e}$. However it is challenging to keep other edge types unchanged and preserve the validity of the tet mesh topology during the collapsing process. In this section, we first present our edge collapse operation for SRF conversion under the assumption that the edge is always collapsible, then discuss how to handle uncollapsible edges.

Collapsible improper singular edge. Under the assumption that improper singular edges are always collapsible, the following algorithm eliminates all of them.

- **Input**: a collapsible improper singular edge ab.
- **Step 1**. For each tet face $\triangle abc$ adjacent to ab, we examine whether both ac and bc are proper singular edges. If yes, we split one of them (see Fig. 3), otherwise directly go to Step 2. This step guarantees that any tet face adjacent to ab does not contain two proper singular edges. It is essential to hold this property otherwise two proper singular edges may be collapsed to an improper singular edge after collapsing edge ab. See more detailed analysis in Appendix C.
- **Step 2**. Collapse edge ab directly. The step reduces the number of improper singular edges at least by one.

The correctness of this algorithm is proved in Appendix C.

Uncollapsible improper singular edge. It is well known that arbitrary edge collapse can make the topology of the tet mesh invalid if *the intersection of the links of two vertices does not equal to the link of the edge* [Dey et al. 1999]. According to the link condition, the uncollapsible improper singular edges can be categorized into two cases.

- **CASE I**. The vertices of an improper singular edge are on the boundary of the tet mesh. The edge cannot be collapsed, otherwise the boundary surface of the tet mesh is changed.
- **CASE II**. For an improper singular edge $\mathbf{pq}$, denote $\mathcal{V}$ as the intersection of the neighboring vertices set of $\mathbf{p}$ and $\mathbf{q}$. $\mathbf{pq}$ is uncollapsible, if there exists a vertex $\mathbf{v}$ in $\mathcal{V}$ such that the triangle $\mathbf{vpq}$ is not a face of any tet. Such a triangle is called *unsuitable triangle*. Figure 4-left illustrates an uncollapsible edge $\mathbf{pq}$ and an unsuitable triangle $\mathbf{vpq}$.

Unlike tetrahedral simplification which does not allow to create new tets, we can convert most uncollapsible improper singular edges of the second case to be collapsible by introducing an edge split operation according to the following proposition.

Proposition 3 *An uncollapsible improper singular edge* $\mathbf{pq}$ *can be converted to be collapsible, if all the unsuitable triangles (if there is any) associated to* $\mathbf{pq}$ *contain at least one non-improper edge. Such a* $\mathbf{pq}$ *can be converted to be collapsible by splitting the non-improper edge of each unsuitable triangle, and the number of improper singular edge is not increased by this operation.*

The constructive proof of Proposition 3 is illustrated in Figure 4.

There is only an extremely rare case that prohibits the above conversion: if an improper singular edge $\mathbf{e}$ is associated to an unsuitable triangle whose three edges are all improper, it is not possible to convert $\mathbf{e}$ to be collapsible since the edge split operation does not reduce the number of improper singular edges. However this extreme case happens only theoretically and we never meet this case in practice. CASE I can be also avoided in practice since we discretize the volume with a high quality and uniform sampled tet mesh by TetGen (http://www.tetgen.org) and the frames on the boundary are relatively smooth due to the way of frame initialization. In our experiments, all the improper singular edges are collapsible by our novel operations.

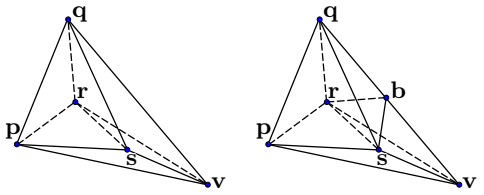


Figure 4: *The left figure shows an uncollapsible edge* $\mathbf{pq}$*. If we collapse it, the face* $\triangle\mathbf{srv}$ *will no longer be shared by any pair of tets. So the topology of the tet mesh is violated. However we can make* $\mathbf{pq}$ *collapsible by splitting edge* $\mathbf{pv}$ *or* $\mathbf{qv}$*. On the right figure, the edge* $\mathbf{qv}$ *is split. Tet* $\mathbf{qrsv}$ *is decomposed into two tets. It is easy to see that collapsing* $\mathbf{pq}$ *will not trigger invalid tet topology. After splitting, we assign the same frame of tet* $\mathbf{qrsv}$ *to tet* $\mathbf{qrsb}$ *and* $\mathbf{vrsb}$*. The new edges* $\mathbf{rb}$ *and* $\mathbf{sb}$ *are nonsingular due to this assignment. Similarly other new edges appearing in the tets around* $\mathbf{qv}$ *are nonsingular. The types of the rest untouched and split edges keep unchanged.*

We demonstrate the matching adjustment and improper singular edge collapse operations on the singular graph in an ellipsoidal volume (see Figure 5). In our experiments, the number of improper singular edges is usually small, thus the quality of the tet mesh is not degraded greatly by edge collapse operation. We notice that a very few tets may have negative volumes due to edge collapse without choosing the middle point of edges carefully, we can further apply tet mesh quality improving techniques [Brewer et al. 2003] to reposition the vertices. In practice all the tets have positive volumes after improvement.

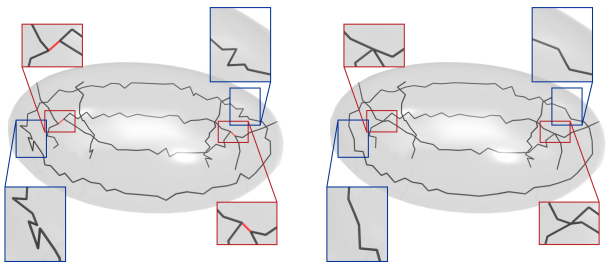


Figure 5: *Left&Right: the singular graph of the frame field in an ellipsoidal volume before and after applying matching adjustment and edge collapse. Improper singular edges (in red) are collapsed (see the red zoomed-in views) and some proper singular edges are straighten by matching adjustment (see the blue zoomed-in views).*

4 SRF-guided Volume Parametrization

Frame guided volume parametrization is an important step to achieve all-hex meshes. Since our SRF-guided volume parametrization is based on the CubeCover approach, we briefly review it here.

A volume parametrization of $\mathcal{V}$ is an atlas of maps $f : \mathcal{V} \to \mathbb{R}^3, p \mapsto (u, v, w)^T$. f is piecewise linear and represented by $f|_{\mathbf{p}}$ on each vertex $\mathbf{p}$ of each tet. The integer grids in $\mathbb{R}^3$ induce a hex tessellation of the volume V.

For two adjacent tets $\mathbf{s}$ and $\mathbf{t}$ with one common face, f can have different values on their shared face but the values need to be related by a transition function so that hex meshing is possible in the parametric domain, i.e.,

$$f|_{\mathbf{t}} = \Pi_{\mathbf{st}} f|_{\mathbf{s}} + g_{\mathbf{st}} \tag{4}$$

where $g_{\mathbf{st}}$ represents an integer translation of the cube grid and is called *gap* in literature. $\Pi_{\mathbf{st}}$ is the matching matrix between the frames defined on $\mathbf{s}$ and $\mathbf{t}$. The gradient field of f: $(\nabla u, \nabla v, \nabla w)$ also defines a piecewise constant frame field over the volume. A good volume parametrization f should align its gradient field to the given 3D frame field as close as possible.

With the frame field and the matchings prepared, it is ready to set up a volume parametrization as introduced in [Nieser et al. 2011]:

$$\min_{f,g} \sum_{\mathbf{t}} vol_{\mathbf{t}} D_{\mathbf{t}} \tag{5}$$

subject to Eqn. 4. Here $D_{\mathbf{t}} = \|h\nabla u_{\mathbf{t}} - \mathbf{U}_{\mathbf{t}}\|^2 + \|h\nabla v_{\mathbf{t}} - \mathbf{V}_{\mathbf{t}}\|^2 + \|h\nabla w_{\mathbf{t}} - \mathbf{W}_{\mathbf{t}}\|^2$, $vol_{\mathbf{t}}$ is the volume of tet $\mathbf{t}$ and h is the length scale of the parametrization set by the user.

However it is non-trivial to ensure a non-degenerate and flipped element-free parametrization. In this section, we first introduce an adaptive rounding algorithm to efficiently solve the volume parametrization and reduce the round-off error in Section 4.1, then study the presence of degenerate elements and propose a novel preprocessing algorithm to reduce degenerate elements greatly in Section 4.2.

4.1 Adaptive rounding algorithm

To extract an all-hex mesh, integer variables, such as the two non-rotational coordinates of the vertices on the singular edges and the gaps $g_{\mathbf{st}}$, should be introduced into the volume parametrization. They can guarantee that the singular edges appear in the hex mesh and the integer iso-lines in the volume are seamless. In addition, for achieving the boundary conforming property, the w-coordinates of each boundary triangular face should be the same integer value (notice that $\mathbf{W}$ directions of the boundary frames are set as surface normals). Therefore, the total number of variables is $12N_c + 3N_{tri}$ with $3N_b + 3N_{tri} + N_s$ being integers. Here N_c denotes the number of tets, N_{tri} denotes the number of interior triangular faces, N_b denotes the number of boundary triangular faces and N_s denotes the number of integer variables associated with the singular vertices.

As pointed out in [Nieser et al. 2011], the integer variables for gap $g_{\mathbf{st}}$ can be eliminated by subtracting two linear equations (4) on each edge to reduce the number of integer variables greatly, i.e.,

$$f|_{\mathbf{t}}(p) - f|_{\mathbf{t}}(q) = \Pi_{\mathbf{st}}(f|_{\mathbf{s}}(p) - f|_{\mathbf{s}}(q)). \tag{6}$$

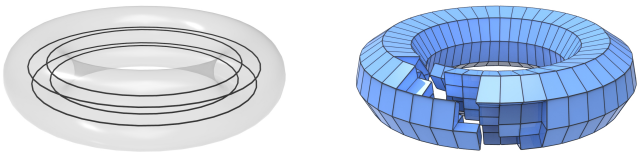


Figure 6: *Left: a torus model with its singular graph. Right: the hex mesh generated from the CubeCover algorithm. The missing hexes are due to non-matched integer gaps.*

The CubeCover approach minimizes Eqn. 5 under the above new constraints by two steps: (1) directly round integer variables; (2) solve non-integer variables. However the direct rounding strategy might introduce large distortion, and the new constraints can induce non-matched integer gaps in some exetreme cases as shown in Figure 6 due to the missing constraints by the above substraction.

We propose to solve the volume parametrization via an adaptive rounding method similar to [Bommes et al. 2009] so as to reduce the distortion and avoid non-matched integer gaps.

Our volume parametrization includes three steps:

1. Compute a global volume parametrization with the reduced linear system without enforcing integer constraints.
2. Adaptively round the integer variables associated with the singular vertices and boundary triangular faces.
3. Compute the integer gaps via adaptive rounding.

We detail each step as follows:

Step 1. To remove the translational freedom of variables, a spanning tree T is first computed in the tet mesh and the gaps between adjacent tets in T are set to 0. We also impose the linear constraints (Eqn. 6) to reduce the number of variables. For each interior triangular face $\triangle pqr$ shared by $\mathbf{s}, \mathbf{t}$: If the gap is 0, the constraint is $f|_{\mathbf{t}} = \Pi_{\mathbf{st}} f|_{\mathbf{s}}$; otherwise, it is

$$f|_{\mathbf{t}}(p) - f|_{\mathbf{t}}(q) = \Pi_{\mathbf{st}}(f|_{\mathbf{s}}(p) - f|_{\mathbf{s}}(q));$$
$$f|_{\mathbf{t}}(p) - f|_{\mathbf{t}}(r) = \Pi_{\mathbf{st}}(f|_{\mathbf{s}}(p) - f|_{\mathbf{s}}(r)).$$

On the boundary triangular face $\triangle pqr$, the constraint is

$$f|_{\mathbf{t}}(p)_w = f|_{\mathbf{t}}(q)_w, \ f|_{\mathbf{t}}(p)_w = f|_{\mathbf{t}}(r)_w.$$

The number of variables can be reduced by applying an integer Gaussian elimination on the collection of these linear constraints C. A continuous solution is then computed by solving the reduced linear system derived from Eqn. 5. This step is the same as the first step of CubeCover [Nieser et al. 2011].

Step 2. We adaptively round all the integer variables to their nearest integers in three steps: (1) If the difference between the continuous solution of the variable X and its nearest integer J is less than 10^{-4}, we fix X to J. If no X is fixed, we only round the integer variable which has the smallest difference. (2) We substitute the fixed variables into the linear constraints C and solve Eqn. 5 by the conjugate gradient method again. Iterate this step until all the integer variables are fixed.

Step 3. Notice that Eqn. 6 does not represent all the constraints from Eqn. 4. The constraint on each interior triangular face $\triangle pqr$ should be added back to complete the original linear constraints:

$$f|_{\mathbf{t}}(p) = \Pi_{\mathbf{st}} f|_{\mathbf{t}}(p) + g_{\mathbf{st}}. \tag{7}$$

Notice that the CubeCover method computes the gaps by direct rounding $g_{\mathbf{st}} := f|_{\mathbf{t}}(p) - \Pi_{\mathbf{st}} f|_{\mathbf{t}}(p)$ without preserving the consistency of linear constraints, thus it may result in non-matched gaps (see Figure 6). We propose to maintain the integer property of the gaps via the above adaptive rounding method. For all the fixed gaps, we append their corresponding linear constraints 7 to C and minimize Eqn. 5. We run this procedure iteratively until all the constraints (Eqn. 7) are appended to C.

4.2 Degenerate elements

Degenerate elements, i.e. the elements with zero volume in the parametric domain, are a big obstacle to hex meshing, since the isolines in the parametric domain will break on the degenerate elements. We first study their presence and then develop topological operations to reduce them.

For a triangular face $\triangle abc$ shared by tets $\mathbf{s}$ and $\mathbf{t}$, it follows by Eqn. 5 in [Nieser et al. 2011]:

$$\begin{cases} f|_{\mathbf{s}}(a) &= \text{type}(\overrightarrow{ab}, \mathbf{s}) \cdot f|_{\mathbf{s}}(a) + g_{ab}; \\ f|_{\mathbf{s}}(b) &= \text{type}(\overrightarrow{ab}, \mathbf{s}) \cdot f|_{\mathbf{s}}(b) + g_{ab}; \\ f|_{\mathbf{s}}(a) &= \text{type}(\overrightarrow{ac}, \mathbf{s}) \cdot f|_{\mathbf{s}}(a) + g_{ac}; \\ f|_{\mathbf{s}}(c) &= \text{type}(\overrightarrow{ac}, \mathbf{s}) \cdot f|_{\mathbf{s}}(c) + g_{ac}; \\ f|_{\mathbf{s}}(b) &= \text{type}(\overrightarrow{bc}, \mathbf{s}) \cdot f|_{\mathbf{s}}(b) + g_{bc}; \\ f|_{\mathbf{s}}(c) &= \text{type}(\overrightarrow{bc}, \mathbf{s}) \cdot f|_{\mathbf{s}}(c) + g_{bc}. \end{cases}$$

Eliminating gaps g_{ab}, g_{ac}, g_{bc} results in a linear system:

$$\begin{cases} (\mathrm{I} - \text{type}(\overrightarrow{ab}, \mathbf{s})) \cdot (f|_{\mathbf{s}}(a) - f|_{\mathbf{s}}(b)) = 0; \\ (\mathrm{I} - \text{type}(\overrightarrow{ac}, \mathbf{s})) \cdot (f|_{\mathbf{s}}(a) - f|_{\mathbf{s}}(c)) = 0; \\ (\mathrm{I} - \text{type}(\overrightarrow{bc}, \mathbf{s})) \cdot (f|_{\mathbf{s}}(b) - f|_{\mathbf{s}}(c)) = 0. \end{cases} \tag{8}$$

Let us start with a simple example. Assume the types of the three edges are

$$\text{type}(\overrightarrow{ab}, \mathbf{s}) = \mathrm{I}; \ \text{type}(\overrightarrow{bc}, \mathbf{s}) = R_x; \ \text{type}(\overrightarrow{ac}, \mathbf{s}) = R_x.$$

By substituting these types into the above linear system, we have $f|_{\mathbf{s}}(a) = f|_{\mathbf{s}}(b) = f|_{\mathbf{s}}(c)$. Thus the parametrization image of $\mathbf{s}$ and $\mathbf{t}$ are degenerate. We apply a similar analysis to all other possible combinations of three edge types and list all the degenerate cases in Appendix D. From the table, we conclude that in the following two cases, the triangular faces are mapped to be degenerate.

- A triangular face contains three singular edges, for instance, a face with edge types $\{R_u, R_v, R_w\}$.
- A triangular face contains only two singular edges whose matching matrices share the same rotation axis, for instance, a face with edge types $\{I, R_v, R_v^3\}$.

Note that if a triangular face contains a nonsingular edge and two singular edges whose matching matrices does not share the same rotation axis, it is not mapped to be degenerate, such as a face with edge types $\{I, R_u, R_v\}$.

In practice a tet can be degenerate due to the integer rounding. For instance, a tet containing three singular vertices but without any singular edge could be degenerate if the rounding procedure makes these singular vertices have the same u-, v- integer coordinates. Another degenerate case happens on the boundary tet where all the four vertices of the tet are on the boundary. Due to the boundary constraints imposed in the parametrization, this kind of boundary tets are mapped to be degenerate always.

So considering the above degenerate cases, we develop two operations : *face-split* and *tet-split*, to handle them.

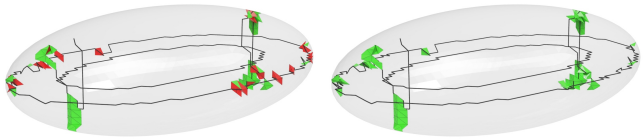
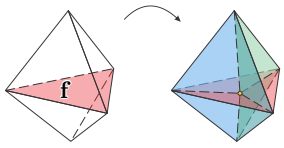
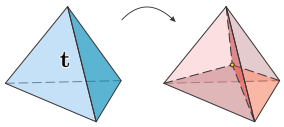


Figure 7: *Left&Right: parametrization without and with our pre-processing. The degenerate tets are rendered in red and the flipped tets are in green. The pre-processing removes all the degenerate tets in parametrization.*



Face-split. For any triangular face $\mathbf{f}$ with more than one singular edge, insert its centroid as a new vertex and split each of its adjacent tets, e.g. $\mathbf{t}$, into 3 tets $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$ (see the inset). The frames of $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$ inherit that of $\mathbf{t}$ so that the new edges are nonsingular. This operation ensures that every triangular face has at most one singular edge.



Tet-split. A tet $\mathbf{t}$ with more than two faces on the volume boundary will be degenerate in the parametrization due to the boundary constraints. We thus split it into four tets $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4$ by inserting its centroid as a new vertex. Again, the frames of $\mathbf{t}_i$ inherit that of $\mathbf{t}$ so that the 4 new edges are nonsingular.

Remark: Notice that our degenerate analysis on triangular faces is only locally done on Eqn. 4 and the integer rounding may still lead to zero-volume tets, we cannot guarantee that the volume parameterizations is degenerate-free after applying these operations. But in practice, most degenerate tets are avoided in the parametrization, for instance, see the example in Figure 7.

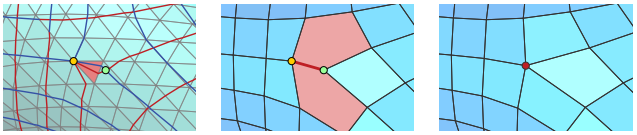
4.3 Flipped Elements

Besides degenerate elements, flipped elements, i.e. the elements with negative volume in the parametric domain, are another obstacle in all-hex meshing. In surface quad meshing, Bommes *et al.* [2009] propose an iterative method to suppress flipped elements by re-weighting the objective function according to the element distortion. However, this method may not converge and cannot guarantee a flip-free parametrization. We tested a similar strategy in our volume parametrization and it was unable to reduce the number of flipped elements significantly. Hence, instead of solving this problem in volumetric parametrization, we leave the processing of flipped elements to the meshing step.

5 All-Hex Mesh Extraction

Given a parameterized tetrahedral volume, the hex elements can be extracted by tracing integer-valued parametric iso-curves in the volume. All-hex meshing can be easily achieved from a flip-free parametrization since the integer iso-curve network admits a valid all-hex structure under this situation. However, the existence of flipped elements may lead to multiple vertices in the iso-curve network which are mapped to the same point in the parametric domain. These vertices result in erroneous edges in the iso-curve network and in consequence introduce non-hex elements. These erroneous edges should be collapsed from the iso-curve network.

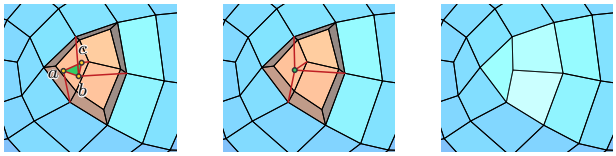
Let us take the surface quad meshing as an example first. The flipped element in Figure 8-a leads to an erroneous edge (the red edge in Figure 8-b) which introduces non-quad faces. However, a simple edge collapse operation can fix the topology of the iso-curve network and restore a complete all-quad mesh (Figure 8-c).



(a) (b) (c)

Figure 8: *(a) An iso-curve network on a parameterized triangular mesh. The red triangle is mapped to a flipped element in the parametrization domain). The yellow vertex and the green vertex are mapped to the same point in the parametrization plane. (b) Two pentagons appear in the curve network if we extract polygonal faces directly. (c) Quad faces can be restored easily by collapsing the erroneous edge.*

For extracting an all-hex mesh from the parameterized volume with flipped tets, the above simple edge collapse strategy can be applied. We list the steps as follows. We first record the parametrization coordinates of each vertex of the iso-curve network. Notice that a vertex x in the iso-curve network may have several parametrization coordinates, for instance, x is a singular vertex. We denote the set of these parametrization coordinates as S_x. Then for any edge pq in the iso-curve network if $S_p \bigcap S_q \neq \emptyset$, we collapse pq to repair the topology of the iso-curve network. A typical example is shown in Figure 9 where three yellow vertices a, b, c satisfy $S_a \bigcap S_b \bigcap S_c \neq \emptyset$. After collapsing edges ab and ac, the missing hexes can be recovered. This simple strategy usually works well when flipped tets are isolated in the volume. But if there exist a large amount of connected flipped tets, the iso-curve network cannot be repaired easily. In Section 7, we show a failure case.



(a) (b) (c)

Figure 9: *(a) Due to the presence of flip tets, it is not possible to recover hexes in the orange region from the iso-curve network since three yellow vertices form a triangular face. (b) Collapse edges among three vertices. (f) The missing hexes are recovered.*

Feature preserving and quality improvement. The generated hex mesh may still suffer large distortion in the vicinity of singularities due to integer rounding. An iterative hex mesh smoothing scheme similar to [Zhang et al. 2005] is thus applied to improve the quality of the hex mesh: (1) Any interior vertex is updated to the centroid of its neighbor hexes. (2) On the hex mesh boundary, any vertex is updated to the centroid of its neighbor quads, and projected back to the input volume boundary. The feature curves are projected on the recognized features of the boundary surface. The quality of the smoothed hex mesh is then further improved by Mesquite software [Brewer et al. 2003].

6 Experiments and Comparisons

Our method is tested on various CAD models and natural shapes on a 2.4GHz Intel Xeon CPU with 12GB of RAM and the results are compared with CubeCover [Nieser et al. 2011] and PolyCube [Gregson et al. 2011]. Statistics and timings are reported in Table 1. The boundary frame fields in our experiments are initialized with principal-dominant cross fields on boundary surfaces to capture the geometry feature of the input shapes. The edge scale h used in the

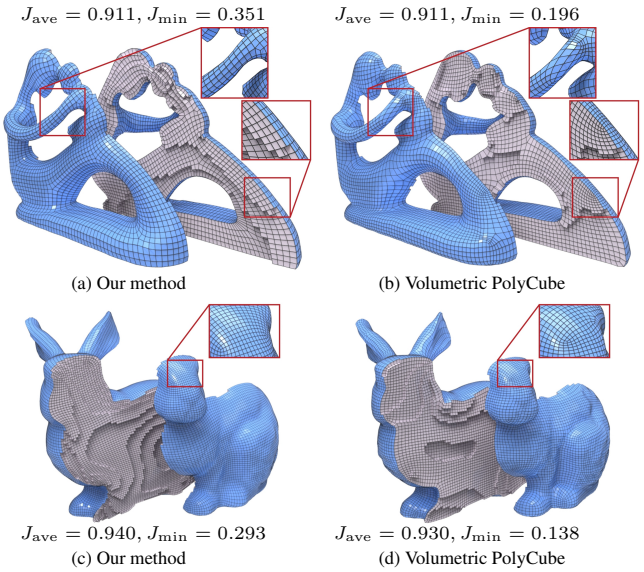


Figure 10: *Comparisons with volumetric PolyCube [Gregson et al. 2011] on fertility and Stanford Bunny. We can see from cut views that our results have more appropriate singularity placement.*

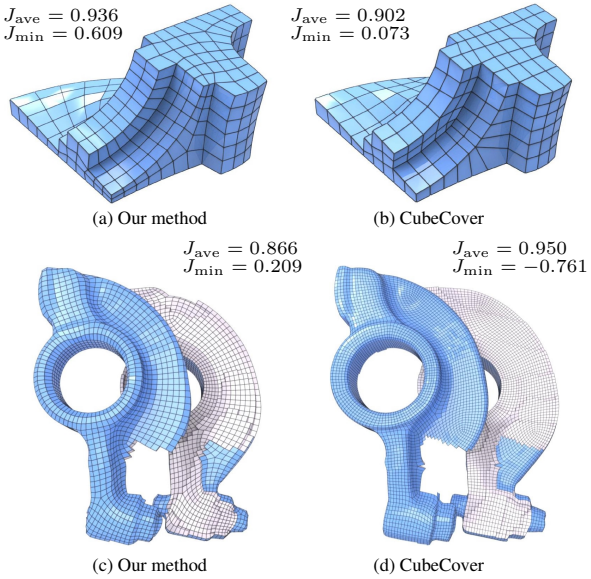


Figure 11: *Comparison with CubeCover [Nieser et al. 2011] on fandisk and rocker arm shown with cut views.*

volume parametrization is the average edge length of the tetrahedral mesh. In all our experimental results, we use orthonormal frames and make the number of hexes generated by our method as close as possible to other methods in comparisons. However, since the number of hexes is affected by edge scale h, types and locations of singularities, our method may produce more hexes. In such cases, we also provide the quality evaluation of the subdivided hexes of other methods (see Table 1). The qualities of hex meshes in this paper are all improved by Mesquite.

Figure 2 shows three meshing results on an ellipsoid initialized with different boundary frame fields. The first and third rows illustrate the corresponding singular graphs of frame fields and singular structures of hex meshes respectively.

Figure 1&10 show that our hex-meshing results have better quality than volumetric PolyCube [Gregson et al. 2011] on the volumes of rod, fertility and Stanford bunny. Since the corners of the PolyCube are the sources of singularities, we can see that there are many irregular vertices on the boundary of the hex meshes from volumetric PolyCube. For instance, volumetric PolyCube introduces more singularities on cylindrical-like regions than our method.

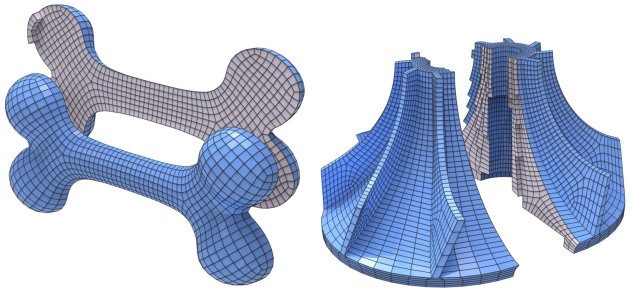
In Figure 11, we compare our results with CubeCover [Nieser et al. 2011]. For the fandisk model, Nieser *et al.* use the coordinate axis direction to guide the frame field. In this way, the resulting mesh is equivalent to using PolyCube which is a single box. Therefore there are only boundary singularities in their result. We initialize the frame field with a feature-aligned cross field, thus the resulting mesh has less distortion due to the interior singularities. For the rocker arm model, Nieser *et al.* create a complex meta-mesh with designed singularities. But the resulting hex mesh still have 17 inverted elements. Our method places singularities automatically in proper positions, thus leads to a hex mesh with better quality.

More hex meshing results are shown in Figure 12. All the hex meshes shown in the paper are provided in the supplemental material in VTK formats.

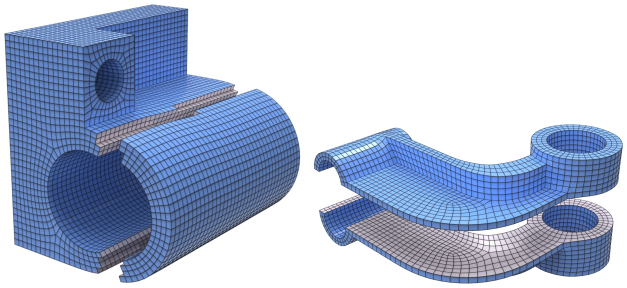
Frame field guiding. We allow the user to guide the frame field generation by placing guiding boxes (Figure 13). The frames of the tets inside the boxes are set to be three local direction vectors of those boxes. The nonsmooth frame transition near to the boundary of boxes can be smoothed out by our frame field optimization. In Figure 2, we show a frame field modified by a guiding box in the right. Another important usage of guiding boxes is to eliminate undesirable singularities of a frame field. In hex meshing, we may not want to introduce many singularities inside some narrow or complex parts of the volume. However, our propagation-based frame field initialization likely generates singular edges around the medial axis of the volume, and most of them cannot be eliminated by frame optimization. We can put guiding boxes at those regions to redefine the frame field there. For instance, we use guiding boxes to modify the frames inside the narrow ears of Bunny (Figure 13-a) and the head of rock arm (Figure 13-b) to reduce singularities. The resulting hex meshes are presented in Figure 10-c and Figure 11-c respectively.

Robustness of SRF conversion. To demonstrate the robustness of our SRF conversion algorithm, we have tried our SRF conversion on a random frame field of an ellipsoidal volume(32320 tets, 6825 vertices). The orthonormal frames in the interior tets are initialized randomly, while the boundary frames are initialized with a principal-dominant cross field and surface normals. The resulting frame field induces 775 proper singular edges and 61 improper singular edges. After SRF conversion, the new frame field induces 753 proper singular edges only. The modified volume contains 31930 tets and 6766 vertices. In the process of SRF conversion, we did not meet any uncollapsible improper singular edges that cannot be handled by our method.

Comparison with [Huang et al. 2011]. Our smoothness function (Eqn. 3) is the summation of the original smoothness measure. However, Huang *et al.* formulate the smoothness energy function as the integral of the gradient of the spherical harmonic coefficients, which is not a direct representation of the original smoothness measure. Their approach may result in a less smooth frame field. We use a simple example to demonstrate the advantage of our method. Figure 14-left shows a singular graph of a spherical volume (194k tets) that is generated by Huang *et al.*' method. The singular graph contains many zigzagged singular curves and two improper singular edges. We further optimize this frame field using our smoothness function and result in a smoother singular graph that does not contain improper singular edges as shown in Figure 14-right.

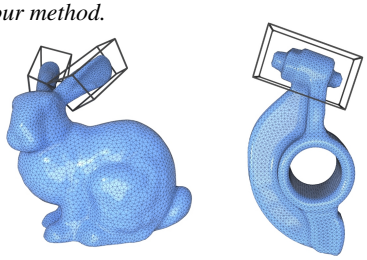


(a) Bone (b) Impeller



(c) Joint (d) Hanger

Figure 12: *Cut views of bone, impeller, joint and hanger hex-meshes generated by our method.*



(a) (b)

Figure 13: *Guiding boxes on the ears of Stanford Bunny and the head of rocker arm.*

7 Conclusion

In this article we present a high-quality all-hex meshing framework based on a SRF guided volume parametrization. We model the smoothness of 3D frame fields efficiently for optimization and present effective topological operations for SRF conversion that is essential to all-hex meshing. We also improve volume parametrization technique by reducing degenerate elements and handling flipped elements in hex mesh extraction.

Limitations. The main drawback of our work is that we have no theoretical guarantee that volume parametrization from an SRF leads to an all-hex structure. The sufficient condition of the existence of all-hex structures requires more deep topological study and is still an open problem for further research. Another theoretical issue is that our SRF representation cannot support high order singularities since integer k used in $\mathcal{R}$ only encodes the fractional part of singularity. Thanks to our initialization method, the high order singularities do not appear in practice. But it is interesting to study how to represent high order singularities. A possible solution is to extend the concept of *period jump* [Ray et al. 2008] to 3D.

As mentioned in Section 5, a large amount of connected flipped elements prohibit all-hex meshing. We show an example in Figure 15 that we are unable to retrieve an all-hex mesh. But notice that the topology of the singular graph does not admit the existence of an all-hex structure, it might explain the appearance

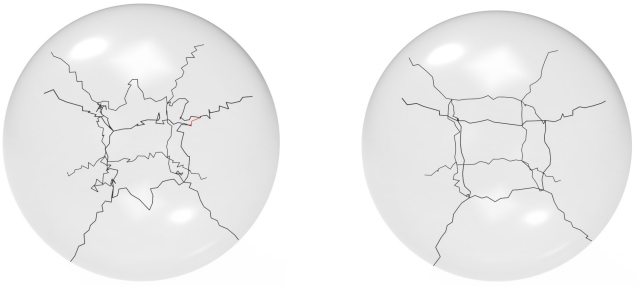


Figure 14: *Left: a singular graph in a spherical volume generated by [Huang et al. 2011]. The red edges are improper singular edges. Right: our optimization result.*

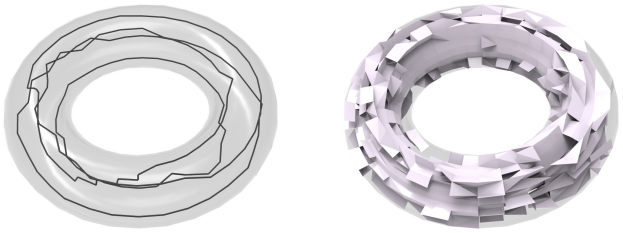
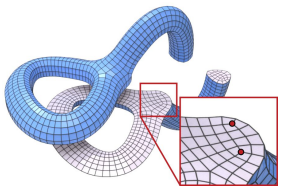


Figure 15: *Left: The singular graph consists of two spiral and close curves inside a torus volume. Right: the tets mapped to negative volumes in the parametrization are rendered.*

of large flipped elements. How to determine the existence of all-hex meshing from the topology of the singular graph is an interesting topic for future research.



Similar to parametrization-based surface quad meshing, our hex meshing method also suffers the misalignment problem (see the right inset) since the global control of singularities is not considered. We plan to generalize the ideas of [Bommes et al. 2011] to solve this problem.

Acknowledgements

The authors would like to thank the reviewers for their valuable comments. Special thanks to Ulrich Reitebuch for providing examples of CubCover, Jin Huang for sharing frame fields, and Steven Lin for proof-reading. The work of Wenping Wang was partially supported by the National Basic Research Program of China(2011CB302400), the Research Grant Council of HongKong (718209, 718010 and 718311).

References

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. (SIGGRAPH) 28*, 77:1–77:10.

BOMMES, D., LEMPFER, T., AND KOBBELT, L. 2011. Global structure optimization of quadrilateral meshes. *Computer Graphics Forum (EuroGraphics) 30*, 375–384.

BREWER, M., DIACHIN, L. F., KNUPP, P., LEURENT, T., AND MELANDER, D. 2003. The mesquite mesh quality improvement toolkit. In *Proc. 12th Inter. Mesh. Roundtable*.

Model	#tet/#hex	Frame Opt.	Solver	Meshing	Scaled Jacobian	Hex Quality	#ce	#sf/#st
ellipsoid (Fig. 2-left)	32k/2088	6.3s	15.6s	0.7s	.950/.752/.049	.777/.367/.117	2	18/20
ellipsoid (Fig. 2-middle)	32k/2178	6.2s	14.1s	0.9s	.918/.715/.050	.720/.327/.107	2	26/25
ellipsoid (Fig. 2-right)	32k/6816	6.6s	16.9s	1.8s	.907/.574/.083	.708/.307/.139	7	28/32
bone (Fig. 12-a)	49k/3396	10.9s	38.1s	2.1s	.930/.620/.067	.757/.175/.142	6	53/61
impeller (Fig. 12-b)	82k/11174	18.2s	276.4s	8.1s	.924/.185/.104	.682/.018/.182	0	2/2
joint (Fig. 12-c)	187k/17784	34.4s	190.4s	9.3s	.984/.729/.026	.913/.488/.071	0	12/16
hanger (Fig. 12-d)	215k/4539	40.7s	305.5s	9.7s	.964/.599/.053	.827/.258/.130	0	62/73
double torus (the inset in Section 7)	44k/272	8.7s	40.0s	1.8s	.891/.717/.067	.716/.408/.115	7	115/135
rod (Fig. 1-c [our method])	41k/5488	7.9s	29.9s	3.5s	.947/.658/.056	.800/.290/.126	0	36/41
rod (Fig. 1-d [Volumetric PolyCube])	-/18028	-	-	-	.950/.131/.076	.820/.004/.157	-	-
fertility (Fig. 10-a [our method])	179k/13584	35.0s	281.4s	19.6s	.911/.351/.088	.693/.034/.154	22	544/610
fertility (Fig. 10-b [Volumetric PolyCube])	-/19870	-	-	-	.911/.196/.100	.712/.025/.176	-	-
bunny (Fig. 10-c [our method])	153k/133632	33.0s	262.1s	56.8s	.940/.293/.079	.762/.057/.146	19	297/345
bunny (Fig. 10-d [Volumetric PolyCube])	-/81637	-	-	-	.930/.138/.084	.747/.011/.164	-	-
subdivided bunny [Volumetric PolyCube]	-/653096	-	-	-	.943/.138/.073	.774/.001/.253	-	-
fandisk (Fig. 1-a,11-a [our method])	22k/357	4.3s	23.6s	0.5s	.936/.609/.063	.734/.184/.130	5	24/27
fandisk (Fig. 1-b,11-b [CubeCover])	-/268	-	-	-	.902/.073/.134	.693/.004/.201	-	-
rocker arm (Fig. 11-c [our method])	254k/10600	46.9s	757.1s	97.7s	.866/.209/.115	.608/.019/.161	16	280/311
rocker arm (Fig. 11-d [CubeCover])	-/35502	-	-	-	.950/-.761/.107	.791/-.082/.158	-	-

Table 1: *Statistics and Timings. We measure the timings of frame field optimization, volume parametrization and mesh generation in seconds. The L-BFGS method used in frame field optimization and the conjugate gradient method used in volume parametrization are parallelized in our implementation. The hex mesh quality is measured by the scaled Jacobian and hex quality [Joe 2008] in the format of average/minimum/standard deviation. #ce is the number of edge collapse operations happened in SRF conversion. #sf and #st are the number of face split operations and tet split operations in handling degenerate elements.*

CARBONERA, C. D., AND SHEPHERD, J. F. 2006. A constructive approach to constrained hexahedral mesh generation. In *Proc. 15th Inter. Mesh. Roundtable*, 435–452.

CRANE, K., DESBRUN, M., AND SCHRÖDER, P. 2010. Trivial connections on discrete surfaces. *Comp. Graph. Forum (Symp. Geom. Proc.) 29*, 1525–1533.

DEY, T. K., EDELSBRUNNER, H., GUHA, S., AND NEKHAYEV, D. V. 1999. Topology preserving edge contraction. *PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE (Beograd) 66*, 80, 23–45.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. C. 2006. Spectral surface quadrangulation. *ACM Trans. Graph. (SIGGRAPH) 25*, 1057–1066.

GREGSON, J., SHEFFER, A., AND ZHANG, E. 2011. All-hex mesh generation via volumetric polycube deformation. *Comp. Graph. Forum (Symp. Geom. Proc.) 30*, 1407–1416.

HAN, S., XIA, J., AND HE, Y. 2010. Hexahedral shell mesh construction via volumetric polycube map. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, 127–136.

HUANG, J., ZHANG, M., MA, J., LIU, X., KOBBELT, L., AND BAO, H. 2008. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph. (SIGGRAPH ASIA) 27*, 147:1–147:9.

HUANG, J., TONG, Y., WEI, H., AND BAO, H. 2011. Boundary aligned smooth 3D cross-frame field. *ACM Trans. Graph. (SIGGRAPH ASIA) 30*, 143:1–143:8.

JOE, B. 2008. Shape measures for quadrilaterals, pyramids, wedges, and hexahedra. Tech. rep.

KÄLBERER, F., MATTHIAS, N., AND POLTHIER, K. 2007. QuadCover — surface parameterization using branched coverings. *Comp. Graph. Forum (Symp. Geom. Proc.) 26*, 375–384.

LIU, D. C., AND NOCEDAL, J. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B 45*, 503–528.

LIU, Y., XU, W., WANG, J., ZHU, L., GUO, B., CHEN, F., AND WANG, G. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph. (SIGGRAPH ASIA) 30*, 140:1–140:10.

MARÉCHAL, L. 2009. Advances in Octree-based all-hexahedral mesh generation: handling sharp features. In *Proc. 18th Inter. Mesh. Roundtable*, 65–84.

NIESER, M., REITEBUCH, U., AND POLTHIER, K. 2011. CubeCover — parameterization of 3D volumes. *Comp. Graph. Forum (Symp. Geom. Proc.) 30*, 1397–1406.

OWEN, S. J. 1998. A survey of unstructured mesh generation technology. In *Proc. 7th Inter. Mesh. Roundtable*, 239–267.

PALACIOS, J., AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. (SIGGRAPH) 26*, 55:1–55:10.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 10:1–10:13.

RAY, N., VALLET, B., ALONSO, L., AND LÉVY, B. 2009. Geometry-aware direction field processing. *ACM Trans. Graph. 29*, 1:1–1:11.

SHEFFER, A., ETZION, M., RAPPOPORT, A., AND BERCOVIER, M. 1999. Hexahedral mesh generation using the embedded Voronoi graph. *Engineering with Computers 15*, 248–262.

SHEPHERD, J. F., AND JOHNSON, C. R. 2008. Hexahedral mesh generation constraints. *Engineering with Computers 24*, 195–213.

SHEPHERD, J., MITCHELL, S. A., KNUPP, P., AND WHITE, D. 2000. Methods for multisweep automation. In *Proc. 9th Inter. Mesh. Roundtable*.

SHIMADA, K. 2006. Current trends and issues in automatic mesh generation. *Computer-Aided Design & Applications 3*, 741–750.

STATEN, M. L., OWEN, S. J., AND BLACKER, T. D. 2005. Unconstrained paving and plastering: a new idea for all hexahedral mesh generation. In *Proc. 14th Inter. Mesh. Roundtable*, 399–416.

TARINI, M., HORMANN, K., CIGNONI, P., AND MONTANI, C. 2004. PolyCube-Maps. *ACM Trans. Graph. (SIGGRAPH) 23*, 853–860.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. In *Symp. Geom. Proc.*, 201–210.

TROTTS, I., HAMANN, B., JOY, K., AND WILEY, D. 1998. Simplification of tetrahedral meshes. In *Visualization '98. Proceedings*, 287–295.

ZHANG, Y., BAJAJ, C., AND XU, G. 2005. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In *Proc. 14th Inter. Mesh. Roundtable*, 449–468.

Appendix

A. The proof of Proposition 1

We shall prove Proposition 1 by contradiction. Assume a singular tet edge $\overrightarrow{\mathbf{pq}}$ ends at interior vertex $\mathbf{p}$ and denote the set of the tets containing $\mathbf{p}$ as S. Since S is homeomorphic to a sphere, all the triangles of S which do not contain $\mathbf{p}$ form a genus-0 triangular mesh M. Select $\mathbf{q}$ as the root node, one can construct a spanning tree G_M on the vertices of M. The complement of G_M is a dual spanning tree called G_M^C. We illustrate G_M and G_M^C in Figure. 16. Since each edge of G_M^C corresponds to two adjacent tets, by traversing G_M^C the frames of all the tets can be reoriented such that the matching matrices of the adjacent tets on G_M^C are the identity. Now we investigate other matching matrices associated with G_M. Geometrically, each leaf edge of G_M connects two adjacent tets and each leaf node of G_M corresponds to a tet edge $\mathbf{e}$ containing $\mathbf{p}$. From the assumption that only the root node of G_M corresponds to the singular tet edge $\overrightarrow{\mathbf{pq}}$, $\mathbf{e}$ is nonsingular obviously. We conclude that the matching matrix associated with each leaf edge must be the identity by Eqn. 2. We can remove all the leaf edges and nodes from G_M, our conclusion is still true for the updated G_M. This removement can be applied on G_M recursively and results that $\overrightarrow{\mathbf{pq}}$ is nonsingular. Thus the assumption is violated. ■

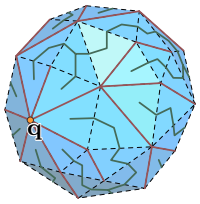


Figure 16: *The spanning tree G_M is in red and the dual spanning tree G_M^C is in green.*

B. Smoothness measure of an orthonormal field

$\sum_{i=1}^{3}\left[H(\widetilde{\mathbf{P}}_{\mathbf{st}}[\cdot,i])+H(\widetilde{\mathbf{P}}_{\mathbf{st}}[i,\cdot])\right] = \frac{1}{512}(564 + 112\cos(2\beta) - 84\cos(4\gamma) - 196\cos(4\beta) - 14\cos(-4\gamma+4\beta) - 14\cos(4\gamma+4\beta) - 56\cos(2\beta-4\gamma) - 56\cos(2\beta+4\gamma) - 84\cos(4\alpha) - \cos(4\alpha+4\gamma-4\beta) - \cos(4\alpha-4\gamma+4\beta) - \cos(4\alpha-4\gamma-4\beta) - \cos(4\alpha+4\gamma+4\beta) + 28\cos(4\alpha-2\beta+4\gamma) + 28\cos(4\alpha+2\beta-4\gamma) + 28\cos(4\alpha-2\beta-4\gamma) + 28\cos(4\alpha+2\beta+4\gamma) - 70\cos(-4\gamma+4\alpha) - 70\cos(4\gamma+4\alpha) - 14\cos(-4\beta+4\alpha) - 14\cos(4\beta+4\alpha) - 56\cos(-2\beta+4\alpha) - 56\cos(2\beta+4\alpha) - 8\sin(4\alpha-3\beta+4\gamma) + 8\sin(4\alpha+3\beta+4\gamma) + 8\sin(4\alpha-3\beta-4\gamma) - 8\sin(4\alpha+3\beta-4\gamma) + 56\sin(4\alpha-\beta+4\gamma) - 56\sin(4\alpha+\beta+4\gamma) - 56\sin(4\alpha-\beta-4\gamma) + 56\sin(4\alpha+\beta-4\gamma))$

C. Edge collapse operation

Under the assumption that improper singular edges are collapsible, we prove that the edge collapse operation is capable to eliminate all improper singular edges.

Assume the edge $\mathbf{e} := \overrightarrow{v_1v_2}$ contained by tet $\mathbf{t}$ is improper. Denote the four opposite tets of $\mathbf{t}$ to the four vertices $\{v_1, v_2, v_3, v_4\}$ by $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4$ respectively and $O(\mathbf{s};\mathbf{e})$ and $N(\mathbf{s};\mathbf{e})$ are the preceding and latter tet of $\mathbf{s}$ around the edge $\mathbf{e}$. We illustrate these tets and vertices in Figure 17. The loops of the neighboring tets of $\overrightarrow{v_4v_3}$, $\overrightarrow{v_1v_4}$, $\overrightarrow{v_2v_4}$ are $\{\mathbf{s}_2, \mathbf{t}, \mathbf{s}_1, \ldots, O(\mathbf{s}_2;\overrightarrow{v_4v_3}), \mathbf{s}_2\}$, $\{\mathbf{t}, \mathbf{s}_3, \ldots, O(\mathbf{s}_2;\overrightarrow{v_1v_4}), \mathbf{s}_2, \mathbf{t}\}$ and $\{\mathbf{t}, \mathbf{s}_1, N(\mathbf{s}_1;\overrightarrow{v_2v_4}), \ldots, \mathbf{s}_3, \mathbf{t}\}$, respectively. We have $\mathrm{type}(\overrightarrow{v_4v_3}, \mathbf{s}_2) = \Pi_{O(\mathbf{s}_2;\overrightarrow{v_4v_3})\mathbf{s}_2} \circ \ldots \circ \Pi_{\mathbf{ts}_1} \circ \Pi_{\mathbf{s}_2\mathbf{t}}$, $\mathrm{type}(\overrightarrow{v_1v_4}, \mathbf{t}) = \Pi_{\mathbf{s}_2\mathbf{t}} \circ \Pi_{O(\mathbf{s}_2;\overrightarrow{v_1v_4})\mathbf{s}_2} \circ \ldots \circ \Pi_{\mathbf{ts}_3}$, $\mathrm{type}(\overrightarrow{v_2v_4}, \mathbf{t}) = \Pi_{\mathbf{s}_3\mathbf{t}} \circ \ldots \circ \Pi_{\mathbf{s}_1 N(\mathbf{s}_1;\overrightarrow{v_2v_4})} \circ \Pi_{\mathbf{ts}_1}$.

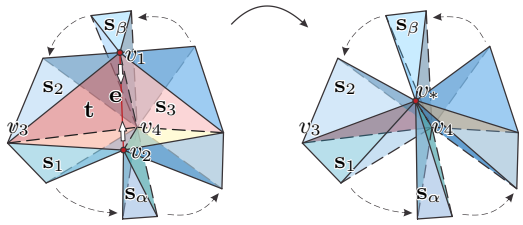


Figure 17: *The tets around $\mathbf{e}$ before and after collapsing $\overrightarrow{v_1v_2}$. v_1 and v_2 are merged to v_*.*

After collapsing $\mathbf{e}$, $\mathrm{type}(\overrightarrow{v_4v_3}, \mathbf{s}_2)$ is $\Pi_{O(\mathbf{s}_2;\overrightarrow{v_4v_3})} \circ \ldots \circ \Pi_{\mathbf{s}_2\mathbf{s}_1}$. Since $\Pi_{\mathbf{s}_2\mathbf{s}_1} = \Pi_{\mathbf{ts}_1} \circ \Pi_{\mathbf{s}_2\mathbf{t}}$ according to our algorithm, the edge type of $\overrightarrow{v_3v_4}$ is not changed. For the new edge $\overrightarrow{v_*v_4}$, its neighboring tets are the union of the original loops of $\overrightarrow{v_1v_4}$ and $\overrightarrow{v_2v_4}$ excluding $\mathbf{t}$ and $\mathbf{s}_3$, i.e., $\{\mathbf{s}_2, \mathbf{s}_1, \ldots, O(\mathbf{s}_2;\overrightarrow{v_*v_4}), \mathbf{s}_2\}$. We have $\mathrm{type}(\overrightarrow{v_*v_4}, \mathbf{s}_2) = \Pi_{O(\mathbf{s}_2;\overrightarrow{v_*v_4})\mathbf{s}_2} \circ \ldots \circ \Pi_{\mathbf{s}_2\mathbf{s}_1}$. Notice that $O(\mathbf{s}_2;\overrightarrow{v_*v_4}) = O(\mathbf{s}_2;\overrightarrow{v_1v_4})$, we can obtain $\mathrm{type}(\overrightarrow{v_1v_4}, \mathbf{t}) \circ \mathrm{type}(\overrightarrow{v_2v_4}, \mathbf{t}) = \Pi_{\mathbf{s}_2\mathbf{t}}^{-1} \circ \mathrm{type}(\overrightarrow{v_*v_4}, \mathbf{s}_2) \circ \Pi_{\mathbf{s}_2\mathbf{t}}$ by a simple calculation. Thus we have

$$\mathrm{type}(\overrightarrow{v_*v_4}, \mathbf{s}_2) = \Pi_{\mathbf{s}_2\mathbf{t}} \circ \mathrm{type}(\overrightarrow{v_1v_4}, \mathbf{t}) \circ \mathrm{type}(\overrightarrow{v_2v_4}, \mathbf{t}) \circ \Pi_{\mathbf{s}_2\mathbf{t}}^{-1}.$$

Now we list the possible edge types of $\overrightarrow{v_*v_4}$.

- If $\overrightarrow{v_1v_4}$ or $\overrightarrow{v_2v_4}$ is improper, $\overrightarrow{v_*v_4}$ could be improper or proper.
- If one of $\overrightarrow{v_1v_4}$ and $\overrightarrow{v_2v_4}$ is nonsingular and another one is proper singular, $\overrightarrow{v_*v_4}$ must be proper. The case can be easily verified by matrix multiplication.

Note that there is no such case that $\mathrm{type}(\overrightarrow{v_1v_4}, \mathbf{t})$ and $\mathrm{type}(\overrightarrow{v_2v_4}, \mathbf{t})$ are proper singular due to the preprocessing in Step 1. From the above analysis, we conclude that, no matter what the type of $\overrightarrow{v_1v_4}$ is, the number of improper singular edges is not increased. This conclusion is also true for all the other affected edges. Since $\mathbf{e}$ is eliminated eventually, the number of improper singular edges decreases at least by one. Thus our algorithm can eliminate all the improper singular edges by applying edge collapse iteratively. ■

D. Degenerate singular combination

Triangular faces with singular edge types in the following table are mapped to be degenerate in the parametrization. Here the numbers $\{0, 1, \ldots, 9\}$ represent the singular types $\{I, R_u, R_u^2, R_u^3, R_v, R_v^2, R_v^3, R_w, R_w^2, R_w^3\}$ respectively. We provide Maple code for reproducing the result in the supplemental material.

(0,1,1)	(0,1,2)	(0,1,3)	(0,2,2)	(0,2,3)	(0,3,3)	(0,4,4)	(0,4,5)	(0,4,6)	(0,5,5)
(0,5,6)	(0,6,6)	(0,7,7)	(0,7,8)	(0,7,9)	(0,8,8)	(0,8,9)	(0,9,9)	(i,j,k)	

Table 2: *Degenerate singular combination. (i, j, k) represents arbitrary combination of three different integers in $\{1, 2, \ldots, 9\}$.*