Automatic Frame Field Guided Hexahedral Mesh Generation

Technique Report

Jin Huang, Tengfei Jiang, Yuanzhen Wang, Yiying Tong, Hujun Bao

State Key Lab of CAD&CG Zhejiang University, Michigan State University

Abstract

We present a hexahedralization method based on a systematic treatment that eradicates the singularities that would lead to degeneration in the volumetric parameterization. Such singularities could be abundant in automatically generated frame fields guiding the interior and boundary layouts of the hexahedra in a purely hexahedral mesh. We first give the mathematical definitions of different types of surface singularities, which are involved in making the surface of the output hexahedral mesh conform to the input model while optimizing mesh quality. We then give a practical framework for adjusting singularity graphs by automatically modifying the rotational transition of frames between charts (cells of a tetrahedral mesh for the volume) to resolve the issues detected in the internal and boundary singularity graph. After applying an additional re-smoothing of the frame field with the modified transition conditions, we cut the volume into a domain with a sphere-like topology automatically, by combining the tetrahedra. Finally, we efficiently solve a mixed integer problem on this domain for a global parameterization, which follows the frame field and respects the transitions across the chart boundaries, and creates a properly connected hexahedral mesh using integer grids of the parameterization.

1 Introduction

Automatic high quality hexahedral mesh generation has sometimes been dubbed the "holy grail" in the meshing community, as such meshes benefit finite element methods due to their tensor product nature, leading to improvement both in speed and in accuracy. Recent development in quadrangulation in computer graphics and geometric modeling has stirred new research effort in this direction based on meshing methods guided by the cross-frame fields, fields of equivalence classes of local frames under the chiral cubic symmetry group (the set of 24 rotations that keep a cube centered at origin invariant without changing the right-handed frame to a left-handed one).

Such methods first create a parameterization of the volume for 3D charts intersecting at common interfaces, followed by extracting the vertices of the hex mesh from the integral points in the parameter domain. The edges of the hexahedra in the mesh would then follow the gradient lines of the parameterization. For computational purposes, the boundary of the resulting mesh must conform to the original boundary and create patches sharing the same integer parameter value over smooth regions, and creating sharp edges and corners near the original features of the mesh. Thus, feature alignment and angle distortion reduction are both linked to a high-quality frame field with one of the axes aligned to the boundary normal on all surface points.

In theory, CubeCover method [Nieser et al., 2011] extending QuadCover method [Kälberer et al., 2007] formulated necessary conditions for the volumetric parameterization, and laid down the foundation for automatic hexahedralization methods based on such parameterizations. They formulated the basic requirements of rotational and translational transitions on each interface between different charts or different parts of the same chart serving as the domains for parameterizations. The nontrivial transitions (cuts) of the domain are necessary to provide flexibility of creating a mesh with high-quality element shape. Otherwise, a polycube-like topology would be enforced, leading to huge angle distortion, undesirable for engineering or scientific computing purposes.

Combining the global volumetric parameterization with compatible rotational and translational transitions as specified in the CubeCover method with automatically computed guidance field, one would imagine that an automatic hexahedral meshing tool would be straightforward to construct. However, unless manually constructed or adjusted, frame fields do not usually satisfy these compatibility conditions for non-degenerate clean volumetric parameterization. There can in fact be a multitude of different types of problems if one computes rotational transitions from automatically generated frame fields that are continuous up to a rotation in the cube symmetry group. The simplest example of the difficulty in such volume parameterization would probably be the case when two edges of a triangle are both along the border of two boundary patches each restricting one parameter to an integer. Thus the triangle will share the same set of two variables, rendering its image in the parameter domain degenerate (collinear). Some other examples include internal line singularities of degenerate types, boundary edges that are required to fold the pairs of triangles adjacent to them onto one another, and nearby singularities forcing the parameters to jump from one integer to another across a short distance.

In this paper, we aim at formulating these problematic cases mathematically and systematically, providing a framework to detect them and treat them in a consistent way, and finally generating pure hexahedral mesh by solving a mixed integer programming problem after automatically cutting the volume into a topologically trivial parameter domain. Our main contributions include:

- We give definitions and condition for checking surface singularities admissible for hexahedral meshing in addition to those for internal singularities.
- We provide a procedure for treatment of the defects in the singularity graph, which if left alone would lead to degenerate parameterization.
- We demonstrate a practical tool for automatical pure hexahedral mesh generation guided by automatically generated guidance cross-frame input fields.

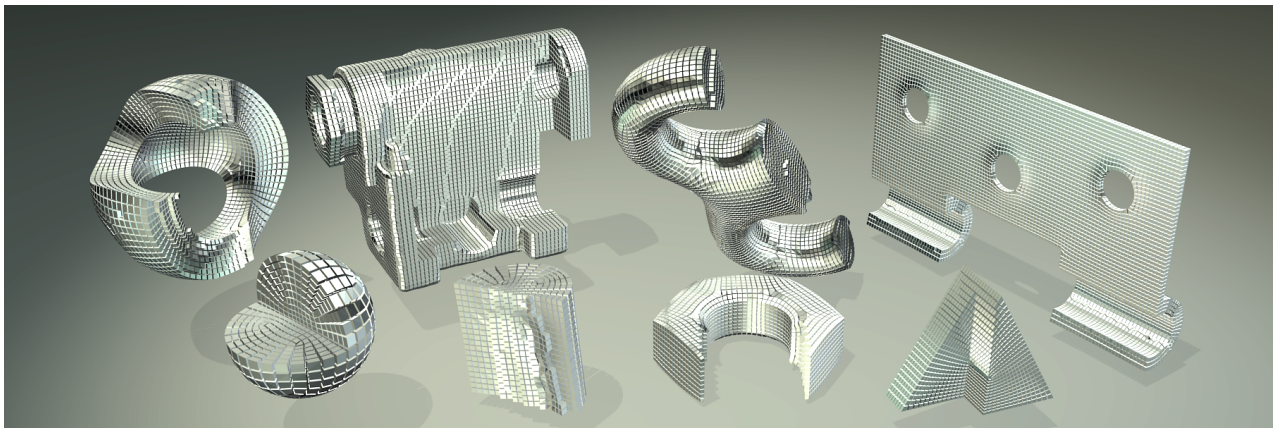

Figure 1: Automatically generated pure hexahedral meshes. Our method is able to handle models ranging from basic primitives to complex ones.

2 Related Work

As mesh generation has been an active research topic for a long period of time, a huge number of methods have been proposed to generate high quality triangular, tetrahedral meshes automatically [Liu et al., 2009, Mullen et al., 2011]. Most of them rely on Delaunay tessellation (dual of Voronoi diagrams), which ensures the connectivity and element quality. However, quadrangular, hexahedral remeshing is substantially more difficult because there is no such dual structure for non-simplicial elements. In 2D manifold quadrangulation, some recent works [Dong et al., 2006, Huang et al., 2008, Zhang et al., 2010] use Morse-Smale Complexes (MSC) [Edelsbrunner et al., 2001] to guarantee a pure quadrangulation based on a scalar function with periodical property. However, 3D MSCs do not necessarily lead to Hexahedral structure. As a consequence, many hexahedral remeshing methods, such as sweeping [Shih and Sakurai, 1996] and paving [Staten et al., 2006] rely heavily on manual input to define the proper topological structure by decomposing the volume into simple parts. Such challenges make grid-based methods [Su et al., 2004] or hex-dominant mesh-based methods [Yamakawa and Shimada, 2003, Lévy and Liu, 2010] often the practical choice for computation performed on automatically generated domains in spite of their inferior quality to hex meshes.

Recent progress shows that global parameterization has achieved great success in surface quadrangulation [Kälberer et al., 2007, Bommes et al., 2009] with the help of a smooth frame field defined on a manifold surface and compatible transition functions between charts. From such a non-degenerate parameterization, a pure quadrangular mesh can be extracted by tracing the iso-lines. To extend such a scheme to hexahedral remeshing, [Nieser et al., 2011] has proposed a global parameterization method for hexahedral remeshing, and [Huang et al., 2011] gives a method to automatically construct a desired frame field. These two works are most relevant to our method. Roughly speaking, we attempt to apply a parameterization method in CubeCover to the automatically generated frame field. However, the task is extremely challenging as the original frame field optimization does not take into account the conditions on admissible singularity types (both inside the volume and on the boundary). In addition, without a coarse meta-mesh manually specified or generated from a manually specified frame field, the generation of a topologically sound partitioning of the volume for a global parameterization is not so straightforward as one might expect. [Gregson et al., 2011] also proposed a method for pure hexahedral remeshing with the topological restriction of no internal singularity, thus leaving few freedom to optimize the shape of hexahedra. As noted in [Nieser et al., 2011], it is often necessary to move the right angle transition line in the parameter domain on smooth regions of the surface into singular edges inside the volume to have better element quality, as the former turns the dihedral angle between a hexahedron's two faces into nearly 180 degrees, while the latter only turns the sum of three such dihedral angles around the internal edge to 360 degrees.

Singularity plays a critical role in detecting and remedying this issue. [Shepherd, 2007] lists many topological restrictions that arise in hexahedral meshing, and uses them in a frame-field-independent remeshing algorithm. The topological constraints in this work apply to the primal elements of a hexahedral mesh, such as node, edge and face etc. Some methods have been proposed to analyze the global topological structure of a quadrangular mesh or a 2D symmetry vector field by singularity graphs [Tong et al., 2006, Palacios and Zhang, 2007]. In 2D quadrangulation, noisy singularity points lead to problematic parameterization results. Thus, some remedies have been proposed for denoising or adjusting the singularities in the frame field [Kälberer et al., 2007]. For hexahedral remeshing, [Nieser et al., 2011] provides a definition on internal singularity types and proved some of its properties. However, there is no existing work on automatic surface singularity and internal singularity adjustment for frame fields used in hexahedral meshing.

3 Overview

Our paper focuses on generating pure hexahedral meshes from input cross-frame fields generated automatically, such as [Huang et al., 2011].

Given a tetrahedral mesh Ω, which is composed of a set of tetrahedra, such that each tetrahedron is associated with a frame $F = \{U, V, W\}$, one member of a cross-frame (an equivalent class of 24 orthonormal frames with axes chosen from $\{U, -U, V, -V, W, -W\}$). To generate a pure hexahedral mesh

with edges following such an input, we compute a parameterization with a method similar to the one proposed in [Nieser et al., 2011]. We loosely follow their notations below.

We denote the parameterization $f = (u, v, w)^T \in \mathbb{R}^3$, which can be expressed in different charts (tetrahedra). In tetrahedron t, we denote its expression by f_t. For two adjacent tetrahedra s and t, the transition from f_s to f_t for a parameterization whose integer grid points can be used in hexahedralization must satisfy

$$f_t = \Pi_{st} f_s + g_{st}, \tag{1}$$

where $g_* \in \mathbb{Z}^3$ and Π_* is one of the 24 permutation matrices for the cross frame containing the standard identity frame (denoted as $[I]$, and we have

$$\Pi_{st} = \Pi_{ts}^{-1},\ g_{st} = -g_{ts}. \tag{2}$$

This requirement is due to the fact that if we use integral lines of the gradients of the parameterization for edges, they must connect to each other across the tetrahedron boundaries. We call any face with a non-identity rotational transition or non-zero translational transition a jump face.

For a surface triangle associated with tetrahedron t, there is a transition Π_t that makes the parameterization coordinates of a surface point p satisfy

$$\Pi_t f_t(p)[0] = g_t, \tag{3}$$

where $g_t \in \mathbb{Z}$ and $\Pi_t \in [I]$. Here, we must have one of the parameters being an integer to avoid cutting the corresponding hexes by the boundary surface.

The translational part g_{st} can be resolved during the parameterization process if the hex edge size can be adjusted, but the rotational part Π_{st} must be properly prescribed so as not to lead to fold-overs and degeneracy in parameterization while following the given frame field.

A natural restriction on the rotational transition is that it should make the angles between the corresponding axes in F_t and $F_s\Pi_{st}$ respectively as small as possible to create a smooth parameterization. If the angle becomes greater than $\pi/2$, the images of the two tetrahedra are highly possible to interfere in the parameterization domain. In addition, more constraints are required to avoid the defects caused by singularities as shown in the next section.

4 Rotational Transition and Singularity

A straight forward method to evaluate the rotational transition is to find the best matching matrix that minimizes the following "alignment error":

$$\begin{aligned} &\|F_t - F_s\Pi_{st}\|, \\ &\|(F_t\Pi_{ta})(1,0,0)^T - n_a\| \end{aligned} \tag{4}$$

where n_a stands for the normal of the surface triangle a in t (there can be more than one boundary face for one boundary tetrahedron). The matrix norm we use in this paper is the Frobenius norm. The rotation for Π_{ta} can be made unique by choosing the rotation fixing one of the axes. Such a setup is necessary for the rotational transition of the gradient fields of the parameterization to be smooth while remaining consistent with the rotational transition of the guidance frame fields [Nieser et al., 2011]. However, it does not guarantee degeneracy-free parameterization results in the presence of certain types of singularities.

4.1 Internal Singularities

If the valence (number of adjacent hexahedral cell) of an internal edge is not 4, it is an internal singularity. As formulated in [Nieser et al., 2011], for an oriented edge e inside of the tetrahedral mesh, which is surrounded by a small counter-clockwisely oriented loop which passes through $(t_0, t_1, \cdots, t_k, t_0)$, define the type of the edge respect to t_0 as:

$$\text{type}(e, t_0) = \Pi_{t_k t_0}\Pi_{t_{k-1} t_k} \cdots \Pi_{t_0 t_1}. \tag{5}$$

If $\text{type}(e, t_0) \neq I$, the edge is a singularity edge. If the type is a rotation around an axis, e.g., X, the parameterization coordinates on the other two axes should be constant integers, i.e. the edge will map to a straight line along the X direction in the (local) parameterization domain. For any other types of singularity type, the edge will be mapped to a point, as there are no eigenvectors of $I - \text{type}(e, t_0)$ producing integer jumps.

The requirement on the inner singularity vertex (Thm 2.2 in [Nieser et al., 2011]) through counting valences would automatically be satisfied when the rotational transitions are computed as the minimization of the frame field transitions.

4.2 Surface Singularities

If the valence of a surface edge is other than 2, it is a boundary singularity edge. Similar to the internal case, for an oriented surface edge e, we also create a small counter-clockwise oriented loop around it, which passes through $(b, x, a, t_0, \cdots, t_k, b)$, where the faces sharing e, a and b, are the triangles adjacent to tetrahedron t_0 and t_k, respectively, and x stands for a point outside of the tetrahedral mesh. We then define the type of the edge as:

$$\text{type}(e, a) = \Pi_{ba}\Pi_{t_k b}\Pi_{t_{k-1} t_k} \cdots \Pi_{t_0 t_1}\Pi_{a t_0}, \tag{6}$$

where Π_{ba} is a rotation around axis U aligning the V, W-axes on triangles a and b when U-axis on each triangle is aligned to the normal. More precisely, it minimizes

$$\|R_e(n_a, n_b)F_{t_0}\Pi_{t_0 a}\Pi_{ab} - F_{t_k}\Pi_{t_k b}\|,$$

where $R_e(n_a, n_b)$ is the rotation around e that aligns n_a to n_b, which flattens the hinge formed by the two boundary triangles. If $\text{type}(e, a)$ is a rotation around an axis by $\pm\pi/2$, it creates a sharp edge on the surface in the parameter domain. If Π_{ab} differs from identity, there is a rotational transition on V, W-axes on the surface. The product of Π_{ab}'s along a loop around a surface vertex can be used to detect singularity vertices on the surface, which forms a sharp corner in the parameter domain with a discrete Gaussian curvature (angle defect) of multiples of $\pi/2$. If the angle defect becomes π, the parameterization will try to wrap two squares around a corner, leading to degeneracy.

The types allowed for boundary edges should be further reduced to 7 admissible cases, excluding the rotations of π around the axes, or there can be extreme angle distortion in the parameterization. Furthermore, a singularity vertex on surface must correspond to a discrete Gaussian curvature (angle defect) $\pm\pi/2$ in the parameter domain.

4.3 Inadmissible Singularities

The following issues will lead to degenerate tetrahedra in the parameterization domain:

- Compound singularity: If the type of a singularity edge is not a rotation around a single axis, we name it a compound singularity. Such compound singularity edges will be mapped to a single point in the parameter domain, thus inducing degenerate tetrahedra. For surface singularities, a rotation around a single axis by π is also treated as a compound singularity.
- Zigzag: If two consecutive singularity edges e_0 and e_1 with the same orientation in a tetrahedron t share the same type, i.e. $\text{type}(e_0, t) = \text{type}(e_1, t)$, these two edges will be mapped to a straight line in the parameter domain, thus making the image of the tetrahedron degenerate.

Thus, we need to schematically adjust the the initial rotational transition set evaluated according to Equation 4 to address the above issues.

In practice, additional care may be necessary to avoid angle distortion. For internal singularity edges, the valences are in most cases less than 6 when the singularity types are generated from the automatically generated guidance frame field. For boundary singularity edges, one may want to make sure that the dihedral angles do not deviate far from the dihedral angles specified by the singularity type in the parameter domain. Similarly, the boundary singularity vertex should have compatible Gaussian curvature in the parameter domain and on the mesh to avoid extreme angle distortion.

4.4 Adjust the Singularities

For an internal face shared by tetrahedra s, t, modifying the rotational transition Π_{st} into $\Pi_{st}\Pi$ by a matching matrix Π on a face will affect the types of its three edges. The type of an edge which is positively oriented to the direction of $s \to t$ will changes to:

$$\text{type}(e, s, \Pi) = \text{type}(e, s)\Pi. \tag{7}$$

In other word, the types of all these edges change by a same rotation. Based on this equation, we use the following operations to remedy the issues in the singularity graph one by one. It can be applied the same way for surface singularities.

To remove a compound singularity (left of Figure 2), we split it into multiple admissible singularities connected to it. First, for an end node x of the compound singularity, we pick two adjacent nodes. One node p is on the correct singularity, and another one q is on the compound singularity. Then, we find a path which connects p, q by non-singularity edges in the one-ring neighborhood of x. These edges and x forms a fan of triangles (in a same orientation). Then we adjust the rotational transitions on these triangles by the same matching matrix to transform the edge $\overline{xq}$ into a regular edge, and at the same time, change the edges in the path into proper singularities. Such an operation can be viewed as moving the singularities $\overline{pxq}$ to the path. The same operation is also used to snap near surface singularities (in the one-ring neighborhood of the surface node) to the surface edges for simpler topological structure.

For a zigzag defect on edges $\overline{px}, \overline{xq}$ (right of Figure 2), we change the rotational transition on the face $\overline{pxq}$ to turn them into regular edges.

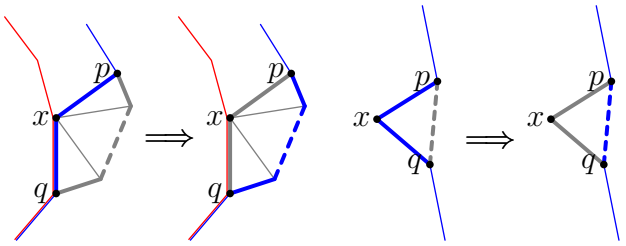

Figure 2: Singularity adjustment. The colored edges are singularity, and the edges in gray are all non-singularity edges. The thick line indicates the edges changed the type in the operation.

In Figure 3, we demonstrate several typical cases in singularity adjustment on a real data. To clearly show the details, we zoom in on parts of the singularities shadowed by colored circles into the right insets surrounded by the same colors respectively. The purple region shows a zigzag on $\overline{pxq}$, and a compound singularity $\overline{pq}$ is highlighted in the blue. The pink region contains more than one issues: $\overline{pq}$ is a compound singularity, we resolve it first by splitting it into two regular singularities $\overline{pq}$ and $\overline{py}$, then the zigzag at $\overline{xpq}$ is shortened into xq.

These operations can be repeatedly performed until all the defects are solved. In our experiments, any order provides enough flexibility to ensure in the end a correct topology in most cases. In the implementation, we fix the compound singularity first, then surface snapping, and finally zigzag. Because adjusting the surface singularity by Π_{ta} never affects the internal singularities, but not vice versa, we clean up all the internal issues before surface ones. The zigzag removing operations are performed last so as to make the surface singularities aligned with the sharp features. In rare cases, not all the defects can be resolved, especially when the tetrahedral mesh is rather coarse.

The adjustment on rotational transition will increase the alignment error, and thus lead to possible difficulties in the following parameterization step. By minimizing the alignment error with respect to F, we can update the frame field according to the new rotational transition. First, we lift the constraints for $F \in SO(3)$, i.e. F is treated as an arbitrary 3 by 3 matrix. Then, using the solution of the linear least square problem in the previous step as the initial value, we include a non-linear penalty of $w\|F^F - I\|^2$ (w is set to 100 in all our results), and solve it again with Gauss-Newton method. The solution will be converted to the best ZYZ Euler angle representation, and serve as the initial value to minimize the alignment error with respect to such a rotation representation, and then transform the solution back to the 3 by 3 matrix representation as the updated frame field. In most cases, the largest angle between the corresponding axes between two adjacent tetrahedra is about 30° when measured with the initial rotational transition, which increases to about 120° after adjusting the rotational translation to resolve compound singularity, and finally decreases to about 75° after updating the frame field.

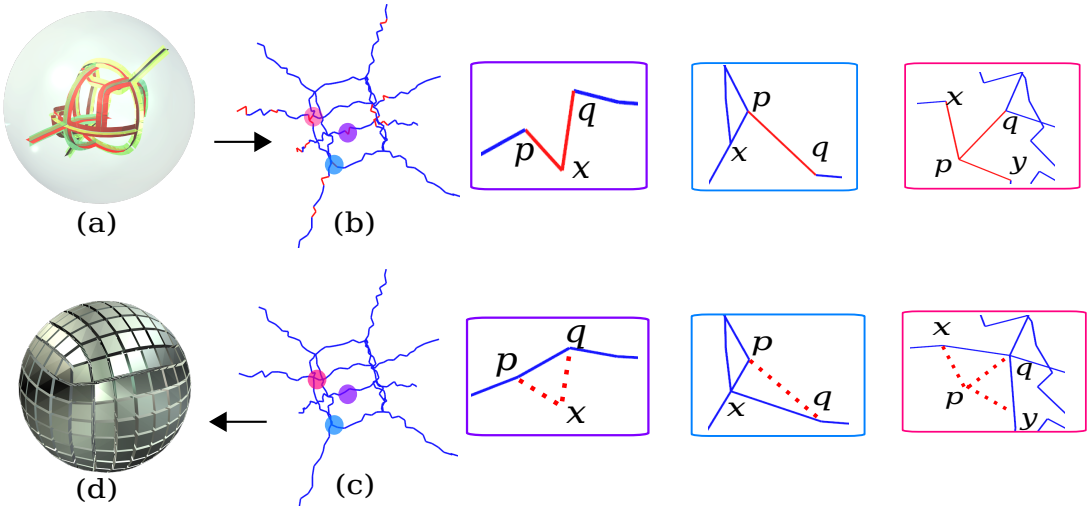

Figure 3: Example on removing inadmissible singularities. (a) is the input frame field visualized by streamline. (b) The initial singularities extracted from it contains many inadmissible singularities (shown in red). After adjusting the singularities into (c), a pure hexahedral mesh (d) can be extracted from the parameterization result.

5 Parameterization

We follow essentially the same procedure in CubeCover method, i.e. finding the minimizer of the following energy

$$\int_V \|\nabla f - F\|^2 \, d\,Vol,$$

which mean that the gradient of the parameterization should follow the axes of the given frame field F as much as possible.

5.1 Topological Simplification

To increase efficiency, instead of simply creating a minimal spanning tree to remove a minimum set of jump variables g across faces between tetrahedra adjacent in the tree. We create a 3D cell with trivial internal topology that fills the inside of the volume to create a proper domain for the global parameterization while removing as many variables associated with faces as possible. Note that the original possibly nontrivial topology is encoded by the self-intersection of the cell's boundary. We use the approach of traversing the volume while maintaining the ball-like topology of a slowly growing cell that eventually take over the whole inside of the boundary surface.

We mark the tetrahedra already inside the cell with a flag. Any non-rotational-jump-face (faces with rotational transition being identity) is a candidate for removal. If such a face is adjacent to tetrahedra both with the flag on, it can only be removed if there is a non-singularity edge adjacent only to this unremoved face. An arbitrary order of traversing such face candidates can leave more faces than necessary. We follow the order of removing the faces around a boundary edge of the growing cell simultaneously to make sure that we remove all non-rotational-jump-faces except for those on at most g membranes necessary to separate the inside of surface with genus-g. This procedure will leave few jump face patches with potentially additional patches representing the translational jumps necessary to cut high-genus models open for a proper global parameterization.

5.2 Parameterization as a Mixed Integer Programming Problem

We can cluster the faces left easily into patches. Each uses only three integer jumps. It is fine to treat each face as an individual face in theory, and use Gauss elimination to reduce the number of variables, or simply leave these variable in the final linear system and increase the number of equations. However, this may turn these constraints into soft constraints as we have to use penalty methods in the minimization process, which will in the end create less smooth transition of the gradient lines. With the patches clearly identified, we can put integer constraints on the internal singularity lines and the boundary patches, and use three floating point values per patch. The approach can greatly reduce the degrees of freedom of the system, and expedite the process of the Gauss elimination if we enforce the hard constraints as in QuadCover method.

The final system normally contains less than few tens of integer variables for a given input frame field. We successively snap one integer variable from the floating point value obtained in the previous solve in a greedy fashion. The linear system is assembled through first removing redundant hard constraints, and then eliminating them from the variables by substitution. This can be done on the gradient operator. After that, the transpose of the reduce gradient operator and the gradient operator can be assembled to get the reduced Laplacian operator for the optimization step. The construction process can take long if the model contains a large number of vertices, but the subsequent linear systems turned out to take little time for each solve when we call the sparse linear solver of Matlab.

We use a simple approach to produce the final mesh. First, we generate the integer points within each tetrahedron, snapping them together through a spatial indexing structure to avoid duplicates. Second, we generate half-integer grid points corresponding to centers of hexes, again snapping them to avoid the case when the centers are located on a face between two tetrahedra. Last, we follow the U,V,W directions to find the corresponding 8 corner vertices for each center. During this process, we keep track of how far it needs to go, and the change of directions when we go across a jump-face.

6 Results

We tested the proposed method on several models. The statistics on the models are given in Table 1 and Table 2. The timing is measured on a PC with an I7 940 CPU at 2.8 GHz and 12 GB RAM. To evaluate the dihedral angles of an edge in a hexahedron, the normal of each adjacent quad face is evaluated by averaging the four triangle normals in two different tessellations of the quad face. As shown in the table below, nearly all the elements are with decent shape quality measures.

model	tet	compound	zigzag	snap	time
prism	91k	0	4	0	0.454s
cylinder	48k	0	16	0	0.388s
nut	65k	0	0	75	0.391s
sphere	80k	3	28	0	3.299s
Pretzel	169k	5	0	711	14.311s
sculpture	105k	0	0	0	0.709s
hinge	434k	0	0	71	2.544s
CAD	530k	0	0	220	6.270s

Table 1: Statistics of the results. The number of input tetrahedra, inadmissible singularity edge, and time used for fixing them are in columns tetrahedron, compound, zigzag and time.

model	time	hex	angle	length	scaled_jac
prism	0.75m	8960	90/3.749	1.321/0.147	0.993/0.017
cylinder	0.58m	5120	89.999/7.164	1.644/0.334	0.976/0.047
nut	1m	6640	89.984/5.749	1.256/0.160	0.985/0.033
sphere	1.5m	7776	89.962/7.531	1.534/0.446	0.974/0.070
Pretzel	1.17m	18997	89.998/12.10	1.549/0.284	0.936/0.101
sculpture	1.3m	10003	89.989/5.736	1.419/0.299	0.984/0.026
hinge	7.1m	41496	89.999/5.119	1.201/0.257	0.988/0.049
CAD	81.5m	57748	89.998/5.912	1.202/0.281	0.984/0.049

Table 2: Statistics of the results. The time used in parametrization is listed in the second column. The following columns list mean/standard deviation of dihedral angles, edge lengths and scaled Jacobian of the parameterization [Shepherd and Tuttle, 2006] for measuring the quality of the output hexahedral mesh.

To better demonstrate the parameterization result with the adjusted singularity, we do not apply any postprocessing to smooth the node position in the hexahedral mesh. Neither do we snap surface node of the output hexahedral mesh to the input model surface nodes. The colored ribbons in the figures visualize the input frame field streamlines along its axes. [Huang et al., 2011].

For models shown in Figure 4, our method naturally capture their symmetry. Even for high genus models, our method is able to automatically find a proper topological structure. In Figure 5, without any manual input, the models Pretzel and Hinge are remeshed into a polycube-like topology, and several internal singularities are automatically introduced on the model sculpture for better quality.

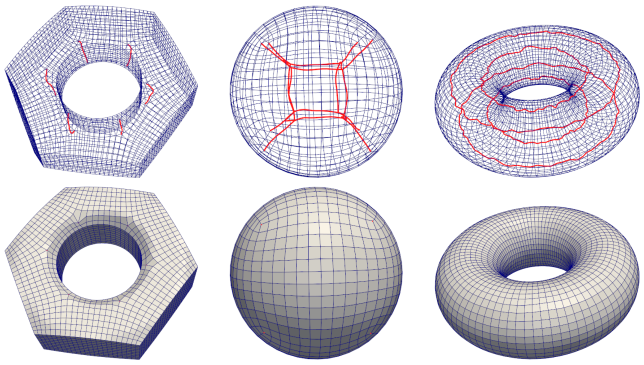

Figure 4: Topological structures of the singularities.

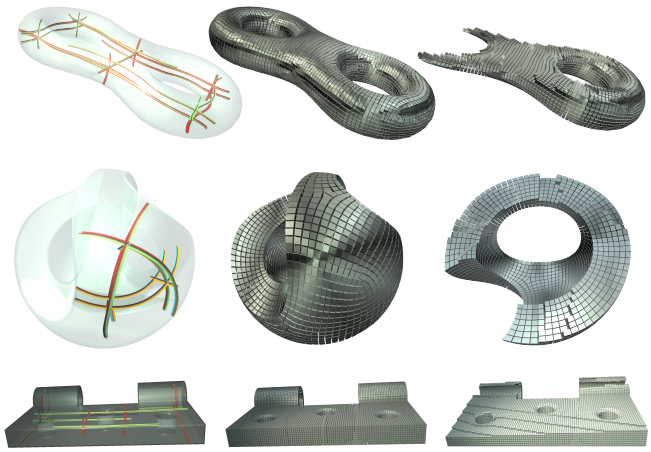

Figure 5: High genus models.

We also tested our method on a very complex CAD model. Manually constructing a meta mesh for such a model can be extremely time-consuming. This model has some small features, which even cause problems in some surface quadrangulation techniques. By choosing a relatively small element size, our method is able to generate a pure hexahedral mesh and preserve important features at the same time. The initial singularity graph extracted from the input frame field has roughly depicted the final connectivity structure. However, the original singularities are on or near the surface, such near misses can lead to degenerate and distorted parameterization. After automatic adjustment, all singularities are snapped to the surface for this model. Subsequently there is no more singularity inside the volume, which renders its topology polycube-like.

7 Conclusion

We present a global volumetric parameterization-based tool to automatically generate purely hexahedral mesh from a 3D frame field, which can be of arbitrary topology as it may be automatically constructed. To eliminate degeneracy in parameterization caused by conflicting rotational transitions often present in an automatically generated frame field, the definition and some analyses on

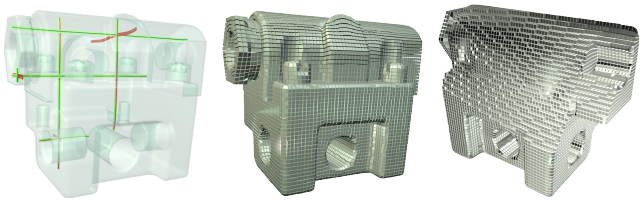

Figure 6: A complex CAD model.

inadmissible internal and surface singularity types are provided in this paper. We also devised a framework to adjust the problematic singularities by applying a sequence of basic operations.

We have no theoretical proof that the set of all the conflicting geometric and topological structure can be detected by using the definition of inadmissible singularity, and thus the method does not provide a sufficient condition to guarantee a complete solution for automatic purely hexahedral remeshing. Indeed, we do find that the sequence of adjustment described in section 4.4 does not always converge to an admissible structure that resolves all the issues in the singularity graph. We conjecture that it may be related to the issues in untangling the topological hex meshes.

We currently focus on the singularity structure. Thus, to get a valid parameterization, we often use a relatively small element size, which leads to a large number of hexahedra. As shown in [Zhang et al., 2010], sizing is important for a more controllable tessellation, and proper sizing can lead to coarser hexahedral mesh. Extension on some recent works [Peng et al., 2011, Tarini et al., 2011] can potentially be employed to coarsen the result into a better mesh.

References

[Bommes et al., 2009] Bommes, D., Zimmer, H., and Kobbelt, L. (2009). Mixed-integer quadrangulation. *ACM Trans. Graph. (SIGGRAPH)*, 28, 3:77:1–77:10.

[Dong et al., 2006] Dong, S., Bremer, P.-T., Garland, M., Pascucci, V., and Hart, J. C. (2006). Spectral surface quadrangulation. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):1057–1066.

[Edelsbrunner et al., 2001] Edelsbrunner, H., Harer, J., and Zomorodian, A. (2001). Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry (SoCG)*, 30(1):87–107.

[Gregson et al., 2011] Gregson, J., Sheffer, A., and Zhang, E. (2011). All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum (SGP)*, 30:5:1407–1416.

[Huang et al., 2011] Huang, J., Tong, Y., Wei, H., and Bao, H. (2011). Boundary aligned smooth 3d cross-frame field. *ACM Trans. Graph.*, 30:143:1–143:8.

[Huang et al., 2008] Huang, J., Zhang, M., Ma, J., Liu, X., Kobbelt, L., and Bao, H. (2008). Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.*, 27:147:1–147:9.

[Kälberer et al., 2007] Kälberer, F., Nieser, M., and Polthier, K. (2007). Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384.

[Lévy and Liu, 2010] Lévy, B. and Liu, Y. (2010). Lp centroidal Voronoi tessellation and its applications. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):119:1–119:11.

[Liu et al., 2009] Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.-M., Lu, L., and Yang, C. (2009). On centroidal voronoi tessellation-energy smoothness and fast computation. *ACM Trans. Graph.*, 28:101:1–101:17.

[Mullen et al., 2011] Mullen, P., Memari, P., de Goes, F., and Desbrun, M. (2011). Hot: Hodge-optimized triangulations. *ACM Trans. Graph.*, 30:103:1–103:12.

[Nieser et al., 2011] Nieser, M., Reitebuch, U., and Polthier, K. (2011). CubeCover - parameterization of 3d volumes. *Computer Graphics Forum (SGP)*, 30:5:1397–1406.

[Palacios and Zhang, 2007] Palacios, J. and Zhang, E. (2007). Rotational symmetry field design on surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 26:3:55.

[Peng et al., 2011] Peng, C.-H., Zhang, E., Kobayashi, Y., and Wonka, P. (2011). Connectivity editing for quadrilateral meshes. *ACM Trans. Graph.*, 30:141:1–141:12.

[Shepherd, 2007] Shepherd, J. (2007). *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, University of Utah.

[Shepherd and Tuttle, 2006] Shepherd, J. F. and Tuttle, C. J. (2006). Quality improvement and feature capture in hexahedral meshes. Technical Report UUSCI-2006-029, The University of Utah.

[Shih and Sakurai, 1996] Shih, B.-Y. and Sakurai, H. (1996). Automated hexahedral mesh generation by swept volume decomposition and recomposition. In *5th International Meshing Roundtable*, pages 273–280.

[Staten et al., 2006] Staten, M. L., Kerr, R. A., Owen, S. J., and Blacker, T. D. (2006). Unconstrained paving and plastering: Progress update. In *In Proceedings, 15th International Meshing Roundtable*, pages 469–486.

[Su et al., 2004] Su, Y., Lee, K., and Kumar, A. S. (2004). Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method. *Computer-Aided Design*, 36(3):203 – 215.

[Tarini et al., 2011] Tarini, M., Puppo, E., Panozzo, D., Pietroni, N., and Cignoni, P. (2011). Simple quad domains for field aligned mesh parametrization. *ACM Trans. Graph.*, 30:142:1–142:12.

[Tong et al., 2006] Tong, Y., Alliez, P., Cohen-Steiner, D., and Desbrun, M. (2006). Designing quadrangulations with discrete harmonic forms. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 201–210, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Yamakawa and Shimada, 2003] Yamakawa, S. and Shimada, K. (2003). Hex-dominant mesh generation with directionality control via packing rectangular solid cells. In *Proceedings of Geometric Modeling and Processing 2002*, pages 2099–2129.

[Zhang et al., 2010] Zhang, M., Huang, J., Liu, X., and Bao, H. (2010). A wave-based anisotropic quadrangulation method. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):118:1–118:8.