Singularity-Constrained Octahedral Fields for Hexahedral Meshing

HENG LIU, RWTH Aachen University
PAUL ZHANG, MIT
EDWARD CHIEN, MIT
JUSTIN SOLOMON, MIT
DAVID BOMMES, RWTH Aachen University

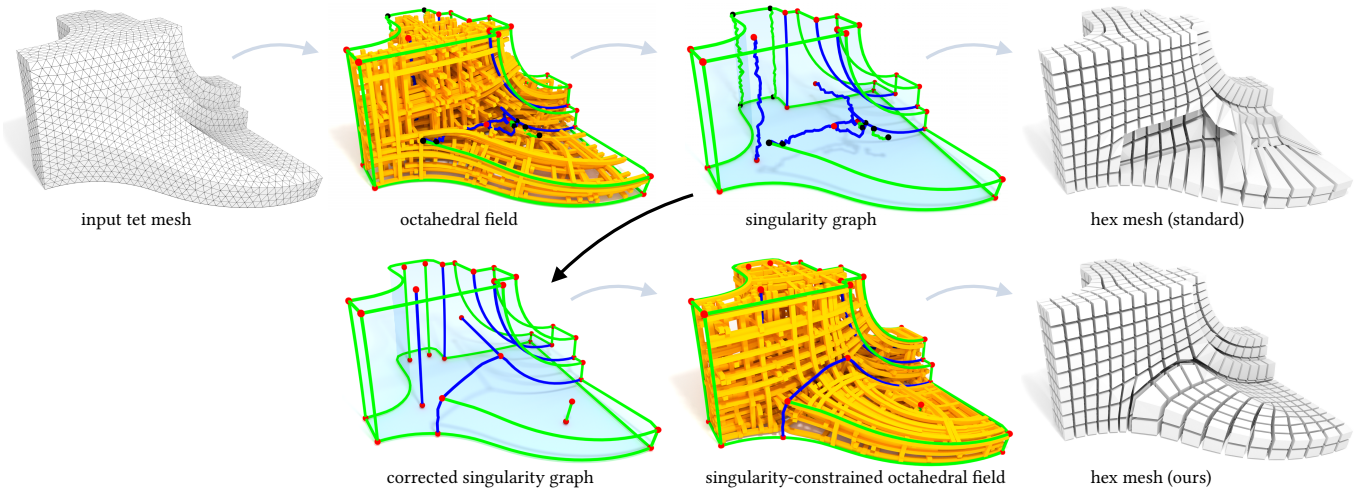


Fig. 1. Overview. Octahedral fields often exhibit singularities that are invalid for hex meshing, inducing non-hex elements and poor quality (top row). Our algorithm (bottom row) starts from a corrected singularity graph and generates a hex-meshable oct. field, resulting in a valid and less distorted hex mesh.

Despite high practical demand, algorithmic hexahedral meshing with guarantees on robustness and quality remains unsolved. A promising direction follows the idea of integer-grid maps, which pull back the Cartesian hexahedral grid formed by integer isoplanes from a parametric domain to a surface-conforming hexahedral mesh of the input object. Since directly optimizing for a high-quality integer-grid map is mathematically challenging, the construction is usually split into two steps: (1) generation of a surface-aligned octahedral field and (2) generation of an integer-grid map that best aligns to the octahedral field. The main robustness issue stems from the fact that smooth octahedral fields frequently exhibit singularity graphs that are not appropriate for hexahedral meshing and induce heavily degenerate integer-grid maps. The first contribution of this work is an enumeration of all local configurations that exist in hex meshes with bounded edge valence, and a generalization of the Hopf-Poincaré formula to octahedral fields, leading to necessary local and global conditions for the hex-meshability of an octahedral field in terms of its singularity graph. The second contribution is a novel algorithm to generate octahedral fields with prescribed hex-meshable singularity graphs, which requires the solution of a large non-linear mixed-integer algebraic system. This algorithm is an important step toward robust automatic hexahedral meshing since it enables the generation of a hex-meshable octahedral field.

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

Additional Key Words and Phrases: hexahedral meshing, octahedral fields, singularity graph, integer-grid maps

ACM Reference Format:
Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. 2018. Singularity-Constrained Octahedral Fields for Hexahedral Meshing. *ACM Trans. Graph.* 37, 4, Article 93 (August 2018), 17 pages. https://doi.org/10.1145/3197517.3201344

1 INTRODUCTION

A key step in physical simulation and physically-based animation involves dividing the spatial domain of the problem into a mesh whose elements are associated to the unknown variables. This discretization procedure has a strong bearing on the stability and fidelity of the result: Poorly-shaped, incorrectly-oriented, or unevenly-sized elements all can lead to failures in the simulation process.

A subtle consideration in meshing involves the *type* of element. In particular, there are many structures into which the domain of the simulation can be divided, including simplicial complexes, rectilinear grids, and even combinations of different types of elements. Among the choices for element type, hexahedral—or cube-shaped—elements are known to yield tight approximation bounds for a simulation despite using fewer elements [Shepherd and Johnson 2008].

Authors' addresses: Heng Liu, RWTH Aachen University, liu@aices.rwth-aachen.de; Paul Zhang, MIT, pzpzpzp1@gmail.com; Edward Chien, MIT, edward.d.chien@gmail.com; Justin Solomon, MIT, jsolomon@mit.edu; David Bommes, RWTH Aachen University, bommes@cs.rwth-aachen.de.

While this suggests that hexahedral (hex) meshing can support more efficient simulation tools, theory and practice currently do not align: few existing hex meshing algorithms provide guarantees on robustness or quality.

Inspired by successful algorithms for two-dimensional quadrilateral meshing, a promising strategy for hex meshing takes place in two steps. First, a set of mutually orthogonal directions is assigned to each point in the volume, with the constraint that one direction aligns to the normal at boundary points; this field—recently termed an *octahedral field* [Solomon et al. 2017]—locally guides the orientations of the hex elements. Next, an integer-grid map is computed with guidance from the octahedral field; pulling back grid lines via this map yields boundary edges of the hexes.

A key challenge in this two-step pipeline involves the topology of the octahedral field. While two-dimensional cross fields only exhibit singular *points* around which the field can circulate, in three dimensions octahedral fields have entire singular *graphs* that determine their topological structure. Certain singularity graphs are incompatible with an integer-grid parameterization, yielding degeneracies or failure at steps downstream in the meshing pipeline. Necessary or sufficient conditions are largely unknown regarding which singular graphs are realizable as hex meshes.

With these issues in mind, our paper presents both theoretical and practical progress toward the overarching objective of topologically-controllable and geometrically-faithful hex meshing. On the theory side, we are among the first to consider the relationship between octahedral field topology and hex mesh topology by enumerating the possible node singularities that can occur given a particular bound on edge valence. Since edge degrees in desirable hex meshes are low, this enumeration comprises a considerable development for the most practical cases of field-guided hex meshing. Additionally, we generalize the Hopf-Poincaré index formula to octahedral fields with locally hex-meshable singularity structure. Beyond our theoretical study, we also present an efficient algorithm to generate octahedral fields with prescribed hex-meshable singularity graphs; this allows admissible singular graphs to be realized as fields that can guide existing hex meshing techniques.

We test our theory and algorithms on several challenging field design test cases. Even for fairly exotic hex-meshable singular graphs, we successfully extract faithful octahedral fields that lead to hex meshes with the desired singularities. The complete source code [Liu et al. 2018] is publicly available as a building block for future research.

2 RELATED WORK

We begin by reviewing state-of-the-art related methods for quad meshing of surfaces, before moving onto related works for hex meshing of volumes. Along the way we highlight some subtle difficulties that arise in this generalization to a volumetric setting.

2.1 Surface Work and Quad Meshing

Field-based parametrization approaches to quad remeshing have been extremely successful in producing high-quality quad meshes. In these approaches, a cross field is used to guide a parametrization of the surface in question to a quantized cone manifold. This allows for a quad mesh to be pulled back onto the surface. We cover some relevant works below and refer to surveys on directional field design [Vaxman et al. 2016] and quad remeshing [Bommes et al. 2013b] for further discussion.

Our technique is heavily influenced by the work of Ray et al. [2008], who produced N-symmetry direction fields with input indices satisfying the Hopf-Poincaré formula. This was achieved by "zippering," or specification of topological matchings in a smart sequential order, which guarantees correct vertex indices. The "chart-zippering" portion of the algorithm described in §5.3 can be understood as a volumetric extension of this strategy with the additional challenge of connecting singularities correctly.

Related work by Crane et al. [2010] achieved a similar goal by directly optimizing the smoothest (continuous) connection rather than initially constructing (discrete) topological matchings. Their approach does not generalize to 3D in a straightforward manner, as rotations do not commute in this setting and $\mathfrak{so}(3)$ is non-abelian. The globally optimal smooth fields of [Knöppel et al. 2013] inspire our method for generating the frame field with fixed topological matchings, described in §5.4.

Full pipelines for quad meshing based on cross fields are proposed e.g. in [Bommes et al. 2013a, 2009] and form the basis of modern quad meshing software. Beyond field design, locally injective quantized parametrization [Campen et al. 2015] and robust mesh extraction [Ebke et al. 2013] techniques are required.

2.2 Hex Meshing

We refer the reader to recent surveys [Armstrong et al. 2015; Yu et al. 2015] for a broad perspective of hexahedral meshing. These surveys cover not only field-based hex meshing but also more classical techniques, e.g. those based on octrees [Marechal 2009] and dual complexes [Erickson 2014; Kremer et al. 2014]. These techniques offer some robustness guarantees but often produce poor topology and/or do not conform well to the geometry of the input domain.

Frame Field Design. Of particular relevance to our field-based approach is the CubeCover algorithm [Nieser et al. 2011]. In this seminal work, the authors introduce the octahedral matchings that are necessary to generate a volumetric parametrization for hex meshing. They prove that the product of matchings about a singular edge must be a rotation about one of the coordinate axes and note a correspondence between local vertex topologies and sphere triangulations. This correspondence immediately yields some restrictions on possible topologies; we extend their observations by enumerating all practically-relevant vertex topologies for the interior and boundary. We also extend their analysis on products of octahedral matchings, showing that partial field information is required to determine the valence associated with a matching product (see §5).

Unlike in the surface case, there is no simple index theory, e.g. via a Hopf-Poincaré formula, to describe the singularity structure of a hex mesh. Thus, Nieser et al. utilize a user-specified coarse hex mesh, termed a *meta-mesh*, to specify their matchings. Our algorithm avoids the burden of designing a coarse hex mesh by starting from a less constrained singularity graph. Completion of our algorithm is a global necessary condition for hex-meshable singularity graphs. Additionally, we generalize the Hopf-Poincaré formula to

the volumetric setting, discussed in §3, giving another global necessary condition on the topology of our input. A discussion of further topological issues is presented in [Viertel et al. 2016] where they note the undesirability of limit cycles, amongst others.

Several works are aimed at producing frame fields, the volumetric analogue of cross fields. The pioneering work [Huang et al. 2011] introduces a representation of frame fields using spherical harmonic coefficients, which is invariant to octahedral symmetries and thus enables the optimization of smooth boundary-aligned fields. [Ray et al. 2016] extends their algorithm with better initialization and boundary conditions; both techniques are further improved in [Solomon et al. 2017] by replacing tet mesh discretization with the boundary element method (BEM). These works are capable of generating smooth octahedral fields but cannot incorporate topological constraints; for this reason their output often is not hex-meshable.

Two works aim to correct input frame fields to produce hex meshes by mesh refinement/coarsening [Jiang et al. 2014; Li et al. 2012], but they are only able to correct certain specific local defects, leaving others unchanged. Additional correction strategies are suggested in [Viertel et al. 2016]. Related work is also present in [Kowalski et al. 2014, 2016] where a vertex-based field is produced and utilized for meshing via sheet tracing, however, without robustness guarantees.

There has also been recent work on design and editing of 3D tensor fields [Palacios et al. 2017], which can represent hexahedral valence 2 and 6. Since the practically most essential cases of valence 3 and 5 cannot be represented, tensor fields are not appealing for hexahedral meshing.

Post-Processing. Once a parametrization is computed, [Lyon et al. 2016] can be used to robustly construct the discrete topology of a hex mesh. After hex mesh extraction, post-processing can improve the quality of a hex mesh. *Base complexes* can be used to progressively simplify hex mesh topology through a series of operations [Gao et al. 2015, 2017b]. Another post-processing approach modifies the frame field induced by an input hex mesh and uses the modified frame field to produce a higher quality hex mesh [Wang et al. 2016]. Lastly, the work of Livesu et al. [2015] performs iterative local improvements on the corners of the hex elements, improving the scaled Jacobians. These methods provide considerable *a posteriori* improvements to a mesh but cannot guarantee conformation to a prescribed singular topology.

Polycube methods. Polycube methods have also been used to generate hex meshes with singular graphs restricted to the boundary surface [Huang et al. 2014; Sokolov and Ray 2015; Tarini et al. 2004]. Polycube parameterizations of the volume can be computed and extruded into the volume to generate nontrivial interior topology [Gregson et al. 2011]. A variation on polycube methods first cuts the volume into ball topology and then designs a frame field on the cut volume to create lower-distortion hex meshes [Fang et al. 2016]. The problem with polycube methods is that they exclude hex mesh topologies that might be required for a low distortion mesh, in particular if internal structures need to be aligned.

Hex-Dominant Meshing. A relaxation of hex meshing is hex-dominant meshing, where the output mesh is allowed to exhibit a small percentage of non-hexahedral elements. Most hex-dominant meshing algorithms relocate vertices of an input tet mesh and then combine as many tets as possible into hexahedra. The placement of the new vertices heavily influences the percentage of the resulting tets that can be made into hexahedra [Baudouin et al. 2014]. Frame field-based approaches have been used to place the new vertices, e.g. [Gao et al. 2017a,a; Sokolov et al. 2016]. Locations where tets cannot be combined into hexahedra can be computed as gap locations and are left as tets [Ray et al. 2017]. Other algorithms for hex-dominant meshing [Bernard et al. 2016; Martin et al. 2012] circumvent the problem of octahedral fields with invalid singularity graphs. While these techniques extract high-quality hex-dominant meshes, there remains a strong demand for pure hex meshes. In particular, simulations and PDEs on hex-dominant meshes can suffer from degeneration of approximation quality near non-hex elements, and specialized finite elements or other techniques are required to faithfully discretize PDEs on these domains.

3 HEXAHEDRAL MESH TOPOLOGY

A hexahedral mesh $\mathcal{H} = (V, E, Q, H)$ is a CW-complex [Hatcher 2001], which decomposes a volumetric region $M \subset \mathbb{R}^3$ into hexahedral cells H, formed by quadrilaterals Q, edges E, and vertices V. The *boundary* $\partial\mathcal{H} \subset (V, E, Q)$ consists of all quadrilaterals ∂Q incident to only one hexahedron, and includes all vertices ∂V and edges ∂E of ∂Q. All remaining mesh elements are referred to as *interior* elements. A hexahedral mesh example is shown in Fig. 2.

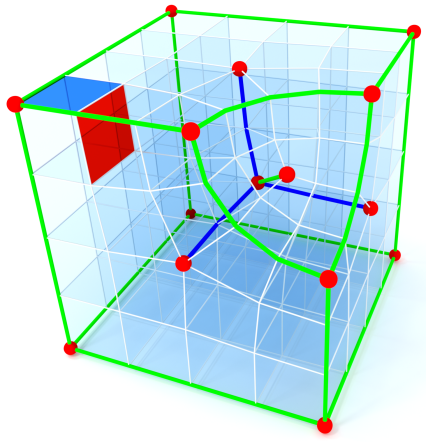


Fig. 2. Hex mesh example. Green and blue edges indicate valence 3 and 5 singularities (valence 1 on boundary). Red vertices indicate singular vertices (interior and boundary). The dark blue and red face indicate two sides of a hexahedron, blue on the boundary, red inside.

Local Topology of Edges. An interior edge is called *regular* if it is incident to exactly four hexahedra; otherwise it is *singular*. Similarly, a boundary edge is regular if incident to exactly two hexahedra, and otherwise it is singular. The *hexahedral valence* $\mathrm{val}_h(\sigma) \in \mathbb{N}_{\geq 1}$ of a mesh element $\sigma \in V \cup E \cup Q$ is the number of its incident hexahedra. Based on this, the *index* $\mathrm{idx}(e)$ precisely specifies the topological

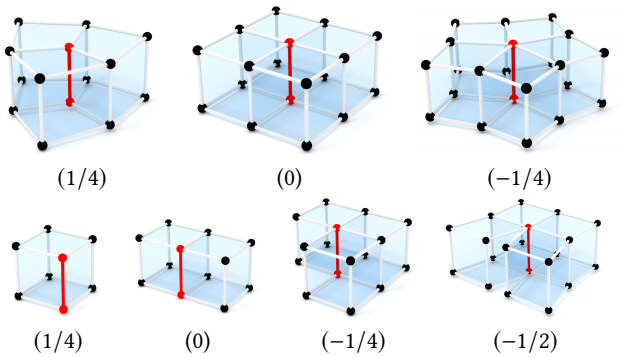


Fig. 3. Practically relevant hex mesh edges, highlighted in red. The top row illustrates interior edges of valence 3, 4, and 5. The bottom row illustrates boundary edges of valence 1, 2, 3, and 4.

type of an edge e by measuring its deviation from regularity:

$$\mathrm{idx}(e) = \begin{cases} (4 - \mathrm{val}_h(e))/4 & \text{for interior } e \\ (2 - \mathrm{val}_h(e))/4 & \text{for boundary } e. \end{cases} \quad (1)$$

For interior edges, this is equivalent to the fractional index of the extruded quad mesh singularity, present in the Hopf-Poincaré formula for cross fields.

In principle, there are infinitely many topological configurations since $\mathrm{val}_h(e) \in \mathbb{N}_{\geq 1}$. Considering the quality of the geometric embedding, however, only a very small subset is practically relevant for hexahedral meshing, as depicted in Fig. 3. For interior edges, only those with hexahedral valence 3, 4 and 5 corresponding to indices from $I_{\text{interior}} = \{1/4, 0, -1/4\}$ are typically desirable since other cases induce lower-quality scaled Jacobians. Moreover, all such low-quality cases could be easily split by sheet insertion [Merkley et al. 2008] into multiple edges of the three good quality cases (cf. Fig. 4). The same argument holds for boundary edges, where the practically important set is given by hexahedral valences of 1, 2, 3 and 4 corresponding to indices $I_{\text{boundary}} = \{1/4, 0, -1/4, -1/2\}$ (cf. Fig. 3). A

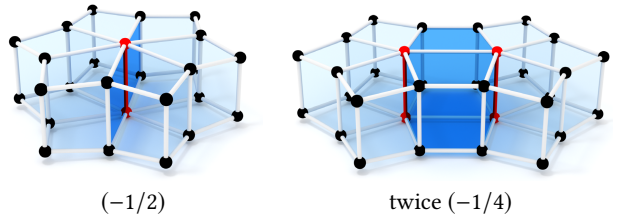


Fig. 4. (Left) Interior singular edge of valence 6 in red. Dark blue faces indicate the location of dual sheet insertion. (Right) Hex mesh after dual sheet insertion. The single valence 6 edge is split into two valence 5 edges.

simply-connected mesh with all interior edges being regular is the pullback of the Cartesian grid under a continuous locally-injective (on the interior) map of the volume into $\mathbb{R}^3$. If this map is also globally injective, then the mesh is homeomorphic to a subset of the Cartesian grid and is often referred to as a *polycube mesh* [Gregson et al. 2011].

Singularity Graph. The subset of singular edges $E_S \subset E$ forms a graph, which can be conformingly partitioned into segments of singular edges with identical index, referred to as *singular arcs*.

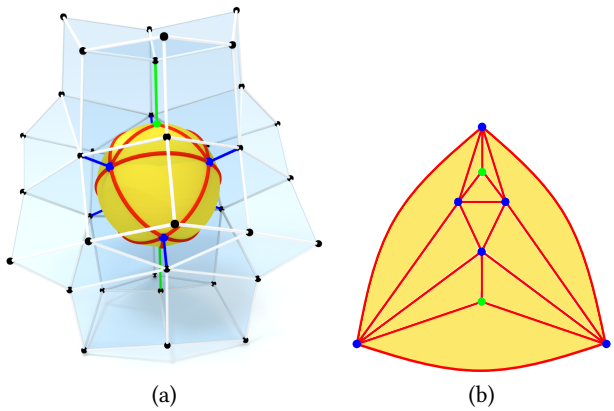


Fig. 5. (a) Interior singular vertex intersected with a yellow sphere. The intersection of the sphere with hex mesh faces is shown in red. (b) Planar representation of the triangulation of the sphere depicted on the left. Vertices correspond to intersections of the sphere with hex mesh edges.

Singular arcs are either closed or bounded by *singular nodes*, where they split into multiple singular arcs or terminate at the boundary. Together, the singular arcs and singular nodes $V_S \subset V$ form the *singularity graph* $\mathcal{S} = (V_S, E_S)$ of the hexahedral mesh (cf. Fig. 2). Tracing sheets from all edges of the singularity graph results in the *base complex* of the hexahedral mesh, which is the coarsest partitioning of the mesh into regular blocks [Gao et al. 2015].

A proposed embedded singularity graph $\mathcal{S} = (V_S, E_S)$ for an input region $M \subset \mathbb{R}^3$ is *globally hex-meshable*, or globally hexable for short, if there is a hex mesh of M with singularity graph matching $\mathcal{S}$. Similarly, we say that $\mathcal{S}$ is *locally hex-meshable*, or locally hexable for short, if there is a hex mesh of the neighborhood of $\mathcal{S}$, with singularity graph containing $\mathcal{S}$ as a subgraph. This is equivalent to having local type assignments for elements of V_S, E_S. For arcs, this is an index, as described above, and for nodes, valid types are described below.

Local Topology of Vertices. A critical theoretical consideration for design of hex meshing algorithms is determining which topological types of vertices exist in a hexahedral mesh. Compared to the simple quarter-integer index for edges the answer is more complicated. Following Nieser et al. [2011], interior vertices of hexahedral meshes topologically correspond to triangulations of the sphere. The idea is to intersect the neighborhood of a hex mesh vertex with a sphere, providing the correspondence of hex mesh edges, faces, and cells with vertices, arcs, and triangular patches on the sphere (see Fig. 5). Consequently, from a topological point of view, hexahedral meshes admit infinitely many different vertex topologies. However, restricting the incident edges to the practically relevant set of hexahedral valences 3, 4 and 5 results in only 11 topologically different interior vertex types. These are illustrated in Fig. 6, and a detailed topological analysis is provided in §A.1.

Similarly, boundary vertices can be classified by intersecting the neighborhood of a hex mesh vertex with a hemisphere, providing the topological equivalence of boundary hexahedral vertices with triangulations of the disc (see Fig. 7). Again, there are infinitely many such triangulations, but we restrict to a practically relevant

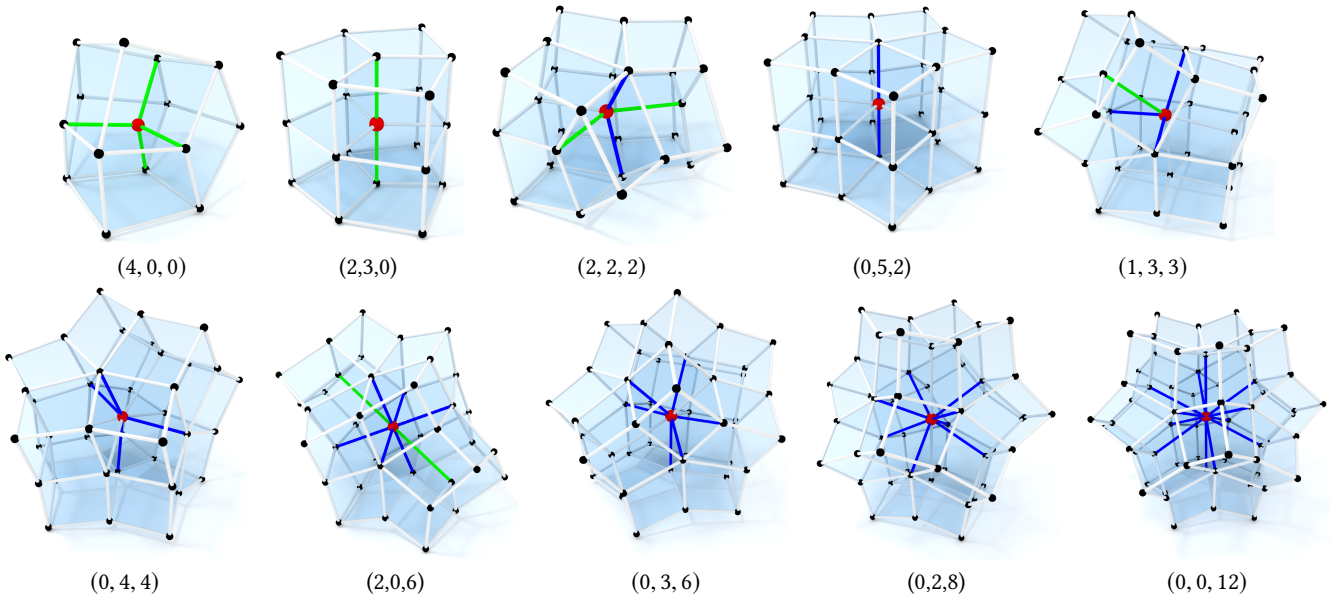


Fig. 6. Singular Node Types. Signatures of the nodes (definition in §A.1) are shown below. Green and blue edges are of valence 3 and 5 respectively.

subset of those with incoming boundary edges of hex valence 1, 2, 3, and 4; incoming interior edges of hex valence 3, 4, and 5; and fewer than 9 incident hexahedra. This results in 237 topologically different singular boundary vertices, which can be found using an exhaustive enumeration algorithm that we describe in §A.2. We use $\mathcal{H}_V$ to denote this set of practically relevant interior and boundary vertex singularity types. A locally hexable valence-$\{3, 4, 5\}$ singularity graph must have all nodes be within this set. This chosen subset of hexahedral vertex topologies coincides with the requirements of many applications but might be inappropriate for others. Unless otherwise noted, our discussions and algorithms are in general not restricted to this choice and could be easily extended to other finite subsets specified by bounds on edge valence and number of incident hexahedra. It is clear that for any application caring about the shape of hexahedra there is an upper bound on the number of hexahedra incident at a vertex, since packing an increasing number of hexahedra into the 4π solid angle of $\mathbb{R}^3$ necessarily deteriorates the worst scaled Jacobian.

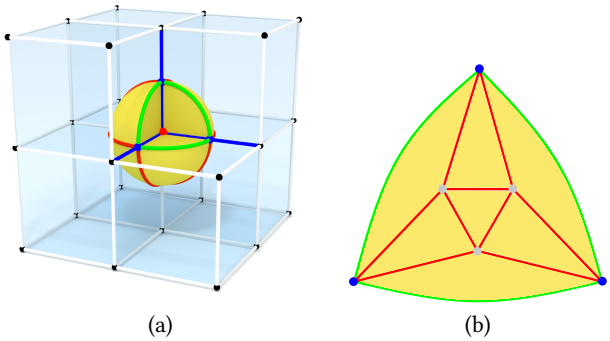


Fig. 7. (a) Boundary singular vertex intersected with a yellow hemisphere. The green boundary of the hemisphere corresponds to intersection with the hex mesh boundary. (b) Triangulation of the hemisphere, with Vertices correspond to intersections between hex mesh edges and the hemisphere.

A Global Necessary Condition. The above classifications define local hexability for a singularity graph. Additionally, we have a global necessary condition for global hexability which is simple to state and check:

$$\sum_{v\in\partial V_S}\frac{1}{2}\left(1-\frac{\mathrm{val}_h(v)}{4}\right)-\sum_{e\in\partial E_S^-}\mathrm{idx}(e) + \sum_{v\in\mathring{V}_S}\left(1-\frac{\mathrm{val}_h(v)}{8}\right)-\sum_{e\in\mathring{E}_S^-}\mathrm{idx}(e)=0, \quad (2)$$

where $\partial V_S, \mathring{V}_S$ denote the boundary and interior nodes of the singularity graph, and $\partial E_S^-, \mathring{E}_S^-$ denote the non-closed boundary and non-closed interior singular arcs. This condition is the analogue of the discrete Hopf-Poincaré formula for quad meshes:

$$\sum_{v\in\partial V}\mathrm{idx}(v)+\sum_{v\in V\setminus\partial V}\mathrm{idx}(v)=\chi(S), \quad (3)$$

where $\mathrm{idx}(v)$ is defined as in Eq. (1) (with substitution of quad valence for hex valence), and S is the quad-meshed surface. Condition (2) can be derived with a simple combinatorial counting argument, given in §B. This condition holds for global hex meshes with arbitrary edge valence, as well as for boundary-aligned locally hex-meshable frame fields—even those not globally hex-meshable. The second generalization is analogous to the fact that Eq. (3) holds for cross fields on surfaces, even if they are not globally quad-meshable. These hex-meshability definitions for fields are discussed in further detail in §4 and a proof for the generalization to locally hex-meshable frame fields is given in supplementary material.

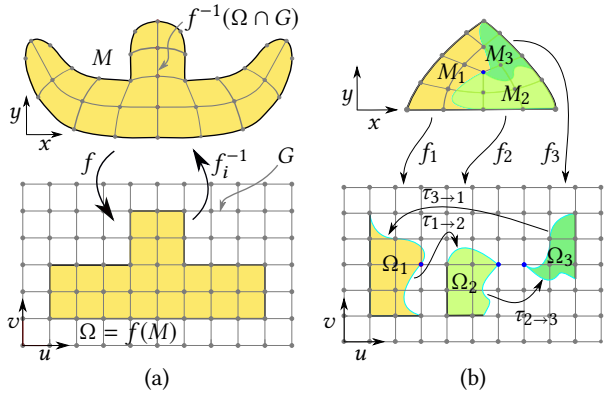


Fig. 8. (a) Polycube map of volume M such that the mapped boundary aligns with the integer grid G and pulled back hexahedra tessellate M. (b) Integer-grid map with three boundary aligned charts $\{(M_1, f_1), (M_2, f_2), (M_3, f_3)\}$ that induce a singular vertex (blue). Transition functions τ define the parametric matching of grid directions.

4 INTEGER-GRID MAPS REVISITED

Polycube Maps. One way to obtain a boundary-aligned hexahedral mesh is to deform the input region $M \subset \mathbb{R}^3$ with a map f that aligns the boundary ∂M with the grid of Cartesian integer isoplanes

$$G = \{(u, v, w)^T \in \mathbb{R}^3 | u \in \mathbb{Z} \text{ or } v \in \mathbb{Z} \text{ or } w \in \mathbb{Z}\}.$$

The decomposition of M into hexahedral cells is then obtained by $f^{-1}(\Omega \cap G)$, which pulls back those hexahedra from G which are covered by the image $\Omega = f(M)$, as illustrated in Fig. 8a. To guarantee a topologically valid hexahedral decomposition, not only boundary alignment but moreover local injectivity of f is required [Bommes et al. 2013a]:

$$\textbf{(Boundary Alignment)} \quad f(p) \in G \qquad \forall p \in \partial M \tag{4}$$

$$\textbf{(Local Injectivity)} \quad \det J_f > 0 \qquad \forall p \in M \tag{5}$$

with $J_f = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z] \in \mathbb{R}^{3\times3}$ being the Jacobian matrix of f. Maps from this class are called *polycube maps* [Tarini et al. 2004]. Unfortunately, only hexahedral meshes without interior singularities can be generated. To achieve the class of all hexahedral meshes, polycube maps must be generalized to *integer-grid maps*.

Integer-Grid Maps. Integer-grid maps generalize polycube maps in a similar way as manifolds generalize simple parametric representations. Imagine that the input region is equipped with an atlas of coordinate charts $A = \{(M_i, f_i)\}$, i.e. a partitioning $M = M_1 \cup M_2 \cup \ldots \cup M_k$ into k parts, each providing its own map $f_i : M_i \to \Omega_i$. To obtain a seamless hexahedral mesh, the piecewise polycube maps $f_i^{-1}(\Omega_i \cap G)$ need to be stitched at all points $p \in M_i \cap M_j$ contained in the intersection of charts. This can be done by restricting the transition functions $\tau_{i\to j} = f_j \circ f_i^{-1}$ between charts i and j to preserve the grid of integer isoplanes, i.e. $\tau(G) = G$. In [Kaelberer et al. 2007; Nieser et al. 2011] it is shown that $\tau(a) = Ra+t$ where $R \in$ Oct is one of 24 orientation-preserving octahedral permutations [Solomon et al. 2017] and $t \in \mathbb{Z}^3$ is an integer translation.

As illustrated in Fig. 8b, the use of charts and grid-conforming transition functions enable the stitching of hexahedral cells in a topologically irregular manner and in particular provides the flexibility to create all types of singularities discussed in §3. Whether a point $p \in M$ is a singularity of the map can be measured by the holonomy of the connection induced by the transition functions. More precisely, if there is any cycle around p with a nonzero holonomy, p belongs to the singularities induced by the map f, denoted by $p \in S_f$. Alternatively, S_f is given by the locus of points where the differential is not well-defined. To guarantee that the conformingly stitched grid consists only of hexahedral cells it is additionally required that all singularities $p \in S_f$ are mapped to the integer grid, leading to the following two additional conditions:

$$\textbf{(Conformity)} \quad \tau_{i\to j}(a) = R_{i\to j}a + t_{i\to j} \qquad \forall a \in f_i^{-1}(M_i \cap M_j) \tag{6}$$

$$\textbf{(Singularities)} \quad f(p) \in G \qquad \forall p \in S_f \tag{7}$$

An atlas of charts $A = \{(M_i, f_i)\}$ where all f_i and $\tau_{i\to j}$ satisfy (boundary alignment), (local injectivity), (conformity), and (singularities) is called an *integer-grid map (IGM)*. Every IGM is guaranteed to induce a boundary-aligned and topology-preserving hexahedral decomposition of M. The reverse also holds: for each hexahedral decomposition, there exists a corresponding IGM.

IGM Induced Frame Fields. Consider one chart (M_0, f_0) of an IGM. Mapping the coordinate frame $[\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}]$ from the parametric domain Ω_0 to M_0 by means of the inverse differential results in a *frame field* $F = J_f^{-1} \in \mathbb{R}^{3\times3}$, which we abbreviate by $F = [d_u, d_v, d_w] \in \mathbb{R}^{3\times3}$. Geometrically, the frame field represents the local orientation of mapped hexahedral elements and is smooth within a single coordinate chart but potentially discontinuous across different charts due to the transition functions. The transformation rule for frames between neighboring charts M_i and M_j is induced by the transition function between Ω_i and Ω_j. It follows from the identity $df_j^{-1}(v) = df_i^{-1}(d\tau_{j\to i}(v))$, meaning that mapping a vector v from Ω_j to M is identical to first transitioning v to Ω_i and then mapping to M (cf. Fig. 8). Considering that $d\tau_{j\to i} = R_{j\to i}$, the resulting rule for transforming a frame F_i from chart i into representation F_j w.r.t. chart M_j consequently is

$$F_j = F_i R_{j\to i} \tag{8}$$

Note the inverse behavior compared to the transformation of a vector $v_i \in T_{\Omega_i}$ in the parametric domain following from Eq. (6):

$$v_j = R_{i\to j} v_i \tag{9}$$

with $R_{i\to j}$ being the inverse of $R_{j\to i}$.

Octahedral Fields. By means of the octahedral group Oct, a frame F extends to its axis set $\mathcal{A}(F) = \{\pm d_u, \pm d_v, \pm d_w\}$, which is a smooth field at all non-singular $p \in M \setminus S_f$, specifically across chart boundaries. This IGM-induced *octahedral field* is orthonormal in the metric of the parametrization. Conversely, a given octahedral field on M can be converted into a frame field by arbitrarily choosing a right-handed subset from the axis set for each chart (from here on frame/octahedral fields are used interchangeably). While every IGM induces a frame/octahedral field, the converse is not true: not every field can be integrated to an IGM. If this is the case, we say the field is *globally hex-meshable*, or globally hexable for short, as the hex mesh given by the integrated IGM has the topology dictated by the

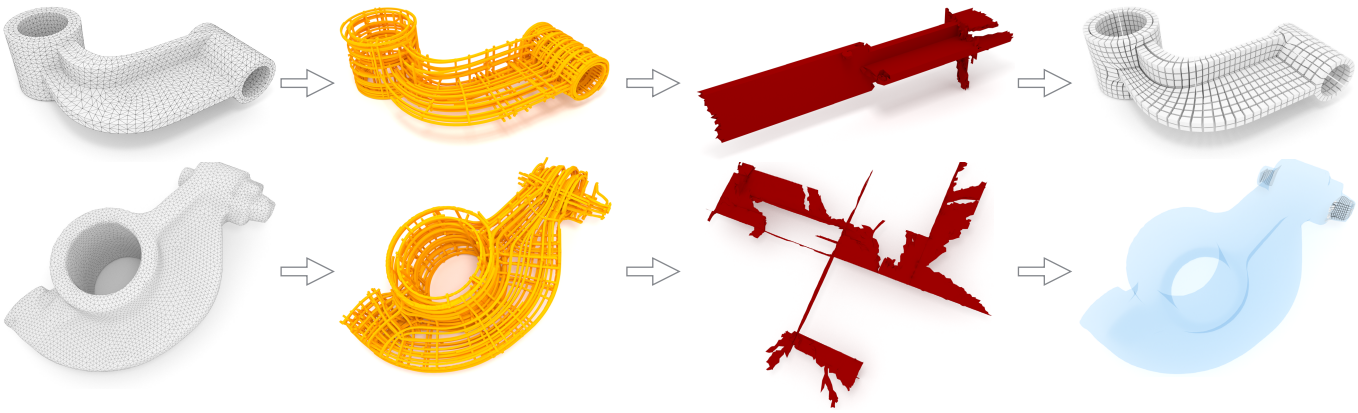


Fig. 9. Top row: splitting approach, an octahedral field (yellow) is generated to guide the IGM parametrization (red). Bottom row: an invalid singularity graph induces constraints leading to a highly degenerate parametrization. Consequently, only very few hexahedra can be extracted [Lyon et al. 2016]

field. In particular, any globally hexable field must have a singularity graph that satisfies the necessary conditions of §3.

IGMs for Hexahedral Meshing. A straightforward hexahedral meshing approach optimizes for a low-distortion map in the class of integer-grid maps. Unfortunately, the resulting optimization problem is extremely challenging due to its high dimensionality, the strong non-convexity due to (5) and especially the huge number of discrete degrees of freedom due to (4), (6) and (7). Consequently, all known IGM-based hexahedral meshing algorithms perform a splitting approach [Jiang et al. 2014; Li et al. 2012], illustrated in Fig. 9. In the first step, a smooth and boundary-aligned octahedral field is generated, e.g. through [Huang et al. 2011; Ray et al. 2016; Solomon et al. 2017], which is then used as a guidance field to find the most similar IGM [Nieser et al. 2011].

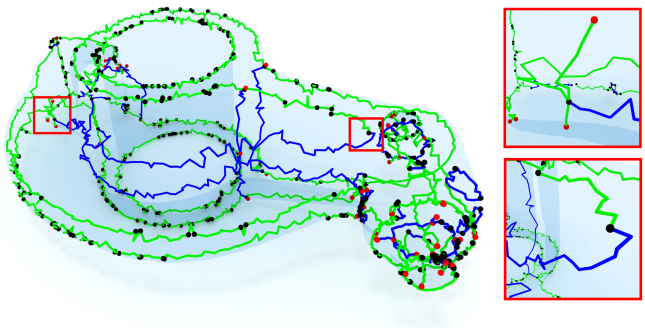


Fig. 10. Typical singularity graph defects of smooth octahedral fields. Green and blue arcs are singularities of index $\frac{1}{4}$ and $-\frac{1}{4}$ respectively. Black spheres indicate invalid node topology, including cases where singular arcs touch the boundary tangentially.

A fundamental problem with this approach is that the singularity graph of the smooth octahedral field is often topologically invalid [Viertel et al. 2016]. As a result, it is common that for seemingly high-quality octahedral fields the IGM still heavily degenerates by violating condition (5), such that in most areas no consistent hexahedral cells can be extracted (cf. Fig. 9, bottom). Typical defects of the singularity graph include local defects, which are singular nodes or arcs of invalid type, or global defects. Common invalid singular nodes that we observed are of type $(3, 0, 1)$ or $(1, 0, 1)$, which are turning points, where a singular arc changes e.g. from valence 3 to valence 5 and returns to its source. An example extracted from a smooth octahedral field is depicted in Fig. 10. Invalid arcs are often created when several arcs in the vicinity of a singular node snap on a common edge, creating a complex type. Such cases are often of local nature and can be resolved by methods like [Jiang et al. 2014; Li et al. 2012]. If an octahedral field has a singularity graph that is locally hexable, then we say the field itself is *locally hex-meshable*, or locally hexable for short.

Even if we have local hexability, the field may still fail to be globally hexable. For example, our global necessary condition stated in §3 might be violated. As another example, there might be no global meshing due to limit cycles (cf. [Sokolov and Ray 2015; Viertel et al. 2016]). The analogue of this problem, occurs even in the surface case, for quad meshing [Campen and Zorin 2017]. The example of a cross-field on a torus with two singularities of index $\pm 1/4$ is shown in this reference. An analogous octahedral field on a thickened torus will also lack global hexability. There is no known characterization of globally hexable fields, or method to generate them. However, the given examples are very specific and locally hexable fields are typically globally hexable and we aim at producing such fields.

Therefore, we advocate a modified splitting approach, including two additional steps: I. repairing the singularity graph and II. generating a new octahedral field, where the corrected singularity graph is preserved, depicted in Fig.1. Our main contribution, presented in §5, is an algorithm for step II, which generates a smooth and boundary-aligned locally hex-meshable octahedral field under the constraint of a prescribed singularity graph. Developing a general solution for step I is left for future work, but nevertheless the set of necessary conditions in §3 is helpful on its own to support correction of singularity graphs. Furthermore, our algorithm can be used for detection of invalid topology and provides information on where inconsistencies are located.

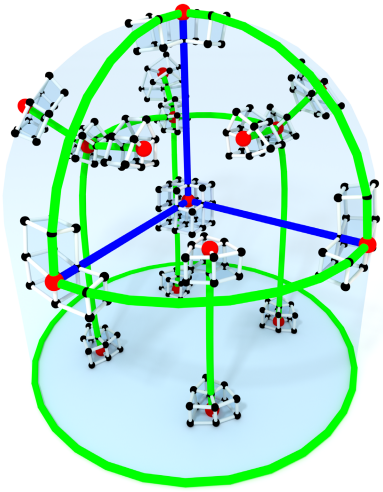


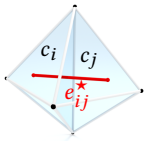
Fig. 11. Visualization of a singularity graph $\mathcal{S}$, used as input for our algorithm. Green and blue curves are singular arcs of index $\frac{1}{4}$ and $-\frac{1}{4}$ respectively. Red spheres are singular nodes and their topological type $\mathcal{S}(v_i) \in \mathcal{H}_V$ is visualized as the corresponding hex-mesh. Note that except for singular edges, $\mathcal{S}$ does not constrain the geometric embedding at nodes.

5 FROM SINGULARITY GRAPH TO OCTAHEDRAL FIELD

Given a tet mesh $\mathcal{T} = (V, E, T, C)$ with vertices V, edges E, triangles T and cells C, and a singularity graph $\mathcal{S}$ embedded in the 1-skeleton $V \cup E$, our goal is to find a discrete octahedral field O that is boundary-aligned and matches the singularity graph $\mathcal{S}$.

Edges and dual edges of $\mathcal{T}$ are equipped with an arbitrary but fixed orientation. Given the end hex-meshing goal, we assume our input $\mathcal{S}$ satisfies the local necessary conditions from §3 and the global necessary condition (2). In particular, $\mathcal{S}$ assigns hexahedral singularity types $\mathcal{S}(e_i) \in \{-1/4, 0, 1/4\}$ to each (oriented) edge $e_i \in E$, and $\mathcal{S}(v_i) \in \mathcal{H}_V$ to each vertex $v_i \in V$. The singular node type $\mathcal{S}(v_i)$ defines the entire local topology of incident singular edges including ordered triplets of singular edges that locally form the corner of a hexahedron. An example of an input singularity graph including singular node topology is shown in Fig. 11.

We assume the octahedral field to be induced by an (unknown) IGM, where each tetrahedron defines its own chart, as discussed in §4. Consequently, the octahedral field $O = (\mathcal{R}, \mathcal{F})$ can be encoded as a set $\mathcal{R}$ of matchings $d\tau_{i\to j} = R_{i\to j} \in \mathrm{Oct}$ for (oriented) dual edges $e^\star_{ij} \in E^\star$ from cell i to cell j, and a set of frames $\mathcal{F}$ with $F_i \in \mathbb{R}^{3\times 3}$ belonging to cell $c_i \in C$.



Splitting the Input Mesh. We require that none of the tetrahedra in $\mathcal{T}$ is adjacent to multiple singular edges or multiple boundary faces. Furthermore, we require that interior singular edges cannot be incident to tetrahedra with a boundary face, that a singular edge cannot be incident to two nodes of the singularity graph, and that a regular edge cannot be adjacent to two singular edges at both vertices. All these requirements can be easily satisfied by a series of edge splits in $\mathcal{T}$. A key property of the resulting mesh is that each singular edge has an independent fan of surrounding incident tetrahedra.

5.1 Singularity Graph Constraints

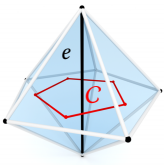
The problem of finding an octahedral field O that exhibits a given singularity graph $\mathcal{S}$ can be formulated as a (nonlinear) algebraic system of constraints for the unknown frames $\mathcal{F}$ and matchings $\mathcal{R}$. The first set of constraints is an immediate consequence of IGMs and ensures alignment of the octahedral field to boundary normals and singular arcs:

(C1) Boundary alignment. At the boundary, the octahedral field aligns to the surface normal, i.e. for each boundary tetrahedron c_i with surface normal n_i and frame F_i we require $n_i \in \mathcal{A}(F_i)$.

(C2) Singular arc alignment. At singular arcs, the octahedral field is tangential, i.e. for each tetrahedron t_i adjacent to a singular edge e_j we require $e_j \in \mathcal{A}(F_i)$.

The second set of constraints, which incorporate matchings $\mathcal{R}$, ensures that for each point in space the octahedral field topology agrees with the topology of the prescribed singularity graph. The topology of the discrete octahedral field is measured in the one-ring tetrahedra incident to an edge or a vertex.

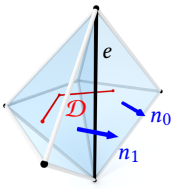
(C3) Edge type. The hexahedral edge type $\mathcal{S}(e)$ specifies the *holonomy* of each dual parametric cycle C, which circles the edge e counter-clockwise in its one-ring neighborhood. Assuming that the dual cycle C starts and ends in tet t with frame F and traverses matchings of dual edges in the order $R_0 \to R_1 \to \ldots \to R_k$, the holonomy is given as the product of these matchings, leading to the condition



$$R_k \ldots R_1 R_0 = \mathrm{rot}(F^{-1}\mathbf{e}, 2\pi\mathcal{S}(e)) \tag{10}$$

where $\mathrm{rot}(\mathbf{a}, \alpha)$ is a rotation around axis $\mathbf{a}$ by angle α. This condition depends on the frame F, which is inevitable since we need to know the parametric coordinate axis to which $\mathbf{e}$ aligns to specify the correct holonomy. More precisely, while the angle $2\pi\mathcal{S}(e)$ is independent of the specific IGM, the rotation axis $F^{-1}\mathbf{e}$ is not, since the IGM could map e to 6 different coordinate axes, each inducing a different constraint.

For edges at the boundary we similarly measure the rotation of the surface normals in the parametric domain by means of an open dual path $\mathcal{D}$ traversing matchings $R_0 \to R_1 \to \ldots \to R_k$ between start frame F_0 and end frame F_1. Since the dual path is open, the surface normals of start and end tetrahedra might correspond to different axes of the coordinate frame, leading to a slightly modified equation

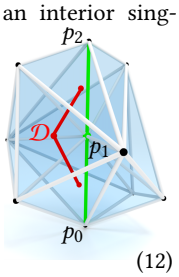


$$R_k \ldots R_1 R_0 F_0^{-1}\,[\mathbf{n}_0, \mathbf{e}] = \mathrm{rot}(F_1^{-1}\mathbf{e}, 2\pi\mathcal{S}(e)) \cdot F_1^{-1}\,[\mathbf{n}_1, \mathbf{e}] \tag{11}$$

Eq. (11) is valid for both interior and boundary edges of any index, since for interior edges $F_0 = F_1$ and $\mathbf{n}_0 = \mathbf{n}_1$ is an arbitrary vector orthogonal to $\mathbf{e}$. Regular edges with $\mathcal{S}(e) = 0$ do not align to coordinate axes but are nevertheless handled correctly since $\mathrm{rot}(\mathbf{a}, 0) = I_{3\times 3}$, independently of $\mathbf{a}$.

(C4) Vertex type. While in the one-ring neighborhood of edges there is only a single rotation degree of freedom, the situation gets more complicated at vertices. The singularity graph defines constraints on the angles in which individual singular edges meet at vertices as well as their precise spatial orientation (e.g. three singular edges form a right-handed corner in frame space). For vertices of the singularity graph an additional set of constraints is required to ensure that all pairwise relations between adjacent singular edges agree with the singularity graph topology. The set of such singular vertex constraints can be decomposed into three different subtypes, the first relating tuples of collinear singular edges, which we call *tangent continuity constraints*, the second relating triples that form a corner in frame space, referred to as *corner constraints*, and the third called *sector constraints* which are the analogue of corners but on the boundary surface.

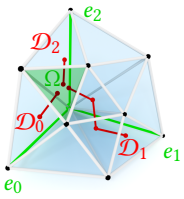
(C4a) Tangent continuity constraint. Along an interior singular arc a consistent axis of the octahedral field is tangential, i.e. no corners or turning points are allowed, where the parametric alignment axis would change. Given a dual path $\mathcal{D}$ connecting two singular edges $\mathbf{e}_0 = \mathbf{p}_1 - \mathbf{p}_0$ and $\mathbf{e}_1 = \mathbf{p}_2 - \mathbf{p}_1$ which are adjacent at a common vertex p_1, tangent continuity means that



$$R_{\mathcal{D}} F_0^{-1} \mathbf{e}_0 = F_1^{-1} \mathbf{e}_1 \tag{12}$$

where $R_{\mathcal{D}}$ is the product of matchings along the dual path $\mathcal{D}$ as before and F_0, F_1 are the frames at start and end of $\mathcal{D}$. The idea is to express the frame axes corresponding to $\mathbf{e}_0$ and $\mathbf{e}_1$ both in the chart of F_1, where they can be compared. Eq. (12) must be satisfied for all dual paths $\mathcal{D}$ that do not enclose other singular arcs.

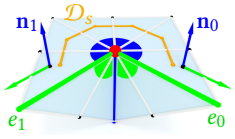
(C4b) Corner constraint. The neighborhood of a singular node can be decomposed into (ordered) triplets of singular edges that form the right-handed corner of a hexahedron, e.g. four such corners for the $(4, 0, 0)$ node type. Each corner corresponds to a triangle in the sphere representation of Fig. 5. Expressing the parametric representation of all three outgoing singular edges in a common chart, results in a right-handed frame inducing the constraint



$$\left[R_{\mathcal{D}_0} F_0^{-1} \mathbf{e}_0 | R_{\mathcal{D}_1} F_1^{-1} \mathbf{e}_1 | R_{\mathcal{D}_2} F_2^{-1} \mathbf{e}_2\right] \in \mathrm{Oct} \tag{13}$$

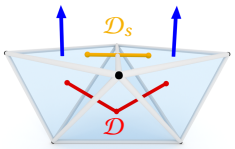
where edge e_i adjacent to frame F_i is transformed along (red) curve $\mathcal{D}_i$ to the (green) common chart Ω for comparison. The constraint should be satisfied for all combinations of curves $\mathcal{D}_i$ inside the region of the corner and not surrounding any singularity. It is enough to satisfy the constraint for a specific set of curves in the region in addition to having zero holonomy on regular edges.

(C4c) Boundary sector constraint. At a boundary singular node, the one-ring neighborhood of surface triangles is partitioned into sectors formed by the incident singular boundary edges. A sector can be convex, flat, concave or a turning point, corresponding to interior angles of $k \cdot \frac{\pi}{2}$ with $k \in 1, 2, 3, 4$ in parametric space respectively. Assuming that a sector is spanned counter-clockwise w.r.t. the surface normal from edge $\mathbf{e}_0$ to edge $\mathbf{e}_1$, which are connected by a dual surface path $\mathcal{D}_s$, a sector constraint is expressed through



$$R_{\mathcal{D}_s} F_0^{-1}[\mathbf{e}_0, \mathbf{n}_0] = \mathrm{rot}\left(F_1^{-1}\mathbf{n}_1, k \cdot \frac{\pi}{2}\right) F_1^{-1}[\mathbf{e}_1, \mathbf{n}_1]. \tag{14}$$

The upper inset figure shows three boundary sectors, adjacent to the central red vertex. Two sectors have an interior angle of $\frac{\pi}{2}$ (green) and one has an interior angle of $\frac{3\pi}{2}$ (blue). Eq. (14) is applied along the yellow path $\mathcal{D}_s$ to match the pairs of normal and edge vectors with the $\frac{3\pi}{2}$ sector angle. This uniquely specifies the rotation $R_{\mathcal{D}}$. In terms of matchings, each dual surface edge of $\mathcal{D}_s$ represents a chain of interior dual path $\mathcal{D}$ as shown in the second inset, i.e. $R_{\mathcal{D}_s} = R_{\mathcal{D}}$.



In summary, octahedral node topology at a tet mesh vertex is fully defined by an overlapping set of corner, sector, and tangent continuity constraints that are induced by the corresponding hex mesh vertex type.

Algebraic system. A singularity graph $\mathcal{S}$ induces the following set of constraints: For each boundary triangle of $\mathcal{T}$ one (C1) constraint, for each edge of $\mathcal{T}$ one constraint of type (C3), if singular additionally one of type (C2), and a set of node constraints (C4) for each vertex of $\mathcal{T}$ which is adjacent to at least one singular edge. The number of (C4) constraints at a vertex depends on the node type described in §3. While some of these constraints may be globally redundant with each other, they are necessary and sufficient for correct local topology.

Finding solutions to the algebraic system is difficult since the number of constraints is large ($\geq 2 \cdot |E|$), the constraints are nonlinear (products in the group of rotations), and the problem involves continuous as well as discrete variables (frames are continuous, matchings and choice of alignment axes are discrete). In the following, we describe a novel algorithm for this task, which based on a careful analysis leverages several structural properties of the algebraic system.

5.2 Decomposition approach

One difficulty of the algebraic system results from the diversity of involved variables (frames and matchings). However, the problem is highly underdetermined, providing flexibility to *a priori* fix alignment of singular edges where required for the solution, leading to a reduced, purely discrete algebraic system only involving the matchings. Once all matchings are determined, the complete set of frames can be found in an independent subsequent step described in §5.4. Our decomposition approach is based on the following observation:

Observation 1. There are $24^{|C|}$ topologically identical representations of a discrete octahedral field with different matchings and frames at faces and cells of a tetrahedral mesh $\mathcal{T}$. This becomes obvious when considering that there are 24 different frames representing one element of an octahedral field and that inversely transforming

a frame and its matchings to neighbors does not alter field topology. Such a topology preserving transformation is simply a change of a local coordinate system.

As an immediate consequence, for every octahedral field there is a choice of coordinate systems, where each singular edge $\mathbf{e}_i$ aligns in all charts of its incident tets t_j to the u-direction, and at the same time each boundary normal $\mathbf{n}_j$ aligns in the chart of its boundary tetrahedron to the v-direction, i.e. the parametric images of singular edges and normal vectors simplify to $F_j^{-1}(\mathbf{e}_i) = \hat{\mathbf{u}}$ and $F_j^{-1}(\mathbf{n}_j) = \hat{\mathbf{v}}$. This is possible since as mentioned before, our tetrahedral mesh is always split such that no tetrahedron is incident to more than one singular edge, and no boundary tetrahedron is incident to an interior singular edge. Our specific choice of coordinate systems greatly simplifies the algebraic system. Conditions (C1) and (C2) are satisfied by construction, and for (C3) all edge holonomies $H_e = \mathrm{rot}(\hat{\mathbf{u}}, 2\pi\mathcal{S}(e))$ are fully determined, simplifying Eq. (11) to

$$R_k \dots R_1 R_0 = H_e \tag{15}$$

with known righthand side and independent of the frames $\mathcal{F}$. In the same way, Eqs. (12), (13) and (14) are simplified to

$$R_{\mathcal{D}}\hat{\mathbf{u}} = \pm\hat{\mathbf{u}}, \tag{16}$$

$$\left[R_{\mathcal{D}_0}\hat{\mathbf{u}} | R_{\mathcal{D}_1}\hat{\mathbf{u}} | R_{\mathcal{D}_2}\hat{\mathbf{u}}\right] \in \mathrm{Oct}, \tag{17}$$

and

$$R_{\mathcal{D}} = \mathrm{rot}\left(\hat{\mathbf{v}}, \pm k \cdot \left(-\frac{\pi}{2}\right)\right), \tag{18}$$

where we exploit the additional convention that the canonical orientation of singular edges is always outward-pointing from nodes of the singularity graph. Choice of the sign in Eqs. (16) and (18) is fully determined by the canonical orientation of singular edges: it is positive if at the common vertex one edge is incoming and the other outgoing, otherwise negative.

In summary, the advantage of our specific choice of coordinate systems is the simplified algebraic system consisting of Eqs. (15), (16), (17), and (18), which solely depends on the matchings $\mathcal{R}$.

5.3 Determining matchings

To find a set of matchings that satisfy the simplified algebraic system, we design a **chart-merging** algorithm. It is based on two principles. (1) In the beginning all matchings are unknown which means that each tetrahedron forms a separate chart. (2) Whenever a matching of a dual edge is determined, we interpret this as merging the charts of both neighboring tetrahedra. The mathematical challenge lies in *locally* determining matchings that are *globally* consistent.

Constrained chart-merging. A frequent operation in our algorithm is to merge two charts while satisfying a constraint on how specific coordinate axes $\mathbf{a} \in \{\pm\hat{\mathbf{u}}, \pm\hat{\mathbf{v}}, \pm\hat{\mathbf{w}}\}$ match along a dual path. More precisely, assume two separate charts A and B are connected by a dual path $\mathcal{D}$ with combined matching $R_{\mathcal{D}} = R_k \dots R_0$. Some of these matchings might already be fixed but since the charts are separate there is at least one R_j that is not yet determined. First, all undetermined matchings on $\mathcal{D}$ other than R_j are set to identity. Then the constrained chart-merging is performed by choosing R_j such that the constraint is satisfied. If the matching of two coordinate axes is specified the combined matching is unique $R_{\mathcal{D}} = R$ and R_j is obtained through

$$R_j = (R_k \dots R_{j+1})^{-1} R (R_{j-1} \dots R_0)^{-1} \tag{19}$$

If matching of only one coordinate axis is specified, i.e. $R_{\mathcal{D}}\mathbf{a} = \mathbf{b}$, we obtain R_j from

$$R_j[(R_{j-1} \dots R_0)\mathbf{a}] = (R_k \dots R_{j+1})^{-1}\mathbf{b} \tag{20}$$

We arbitrarily choose one of the four $R_j \in \mathrm{Oct}$ which maps $(R_{j-1} \dots R_0)\mathbf{a}$ to $(R_k \dots R_{j+1})^{-1}\mathbf{b}$. Finally, if no matching of coordinate axes is constrained, each $R_j \in \mathrm{Oct}$ is a valid solution and we always pick the identity.

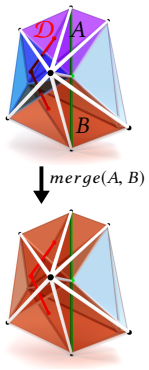


Chart-zippering. The second central operation besides chart-merging is chart-zippering. Given an edge e with all but one matching of incident triangles known, the unknown matching R_j can be uniquely determined based on the edge holonomy Eq. (15):

$$R_j = (R_k \dots R_{j+1})^{-1} H_e (R_{j-1} \dots R_0)^{-1} \tag{21}$$

The operation is called zippering in analogy to the zippering algorithm of [Ray et al. 2008] for N-symmetry fields on surfaces with prescribed singularities. In our terms, the algorithm of [Ray et al. 2008] creates one initial connected chart through a dual spanning tree of surface triangles and then iteratively performs chart-zippering until all matchings are determined. In contrast to the simpler surface case we need to cope with (C4) node constraints which render a simple spanning tree construction infeasible and require a carefully designed chart-merging.

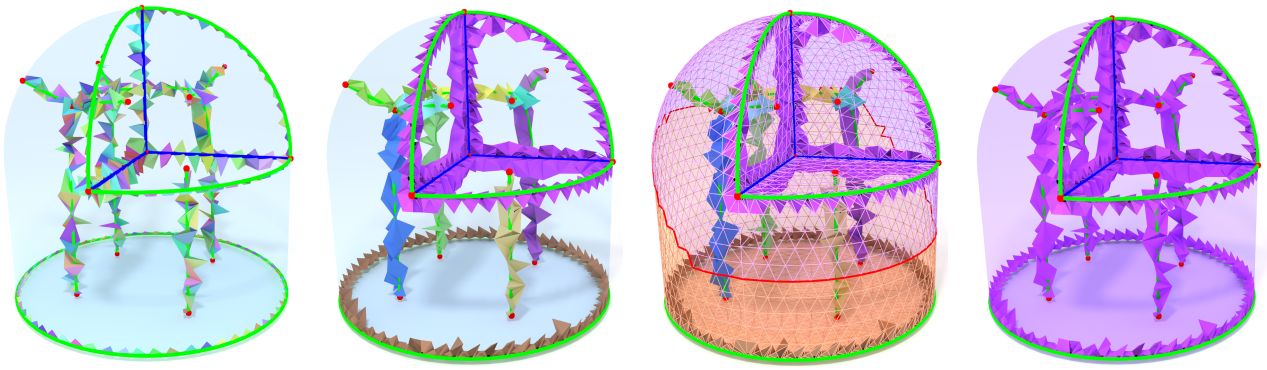
Algorithm. The algorithm performs chart-merging in three major steps outlined in Algorithm 1, to determine the set of matchings X. The idea is to proceed from singular arcs and boundary surfaces, where most information in the form of constraints is available, to the interior of the volume. First charts incident to singular edges are merged along singular arcs, leading to a set of *singular tubes* illustrated in Fig. 12b. In the second step the charts of boundary tets are merged, Fig. 12c. Finally, the algorithm extends the information to the interior until all matchings are determined, Fig. 12d.

Algorithm 1 DetermineMatchings

Input: Tet mesh with singular topology constraints (15)-(18)
Output: matchings $X \in \mathrm{Oct}^{|T|}$ satisfying (15)-(18) or INFEASIBLE

```
X ← flag all matchings as uninitialized
X ← MergeSingularArcCharts(X)        ▷ Singular tube charts
X ← MergeBoundaryCharts(X)           ▷ Boundary shell charts
return MergeVolumeChart(X)           ▷ Solve remaining matchings
```

Step 1: Merging charts of singular arcs. We begin with generating one combined chart for each singular edge. For one singular edge, the set of incident matchings $R_0 \dots R_k$ needs to satisfy the holonomy constraint Eq. (15). Since singular edges are consistently oriented in all incident charts, it is sufficient to set $R_1 \dots R_k$ to identity and determine R_0 through chart-zippering Eq. (21). An example of the resulting singular edge charts is depicted in Fig. 12a. Next we connect charts along singular arcs. For interior singular arcs,

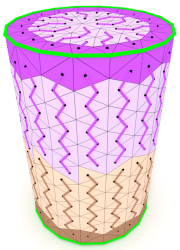


(a) (b) (c) (d)

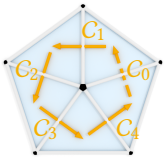
Fig. 12. Algorithm Overview: (a) A separate initial region is created for each singular edge. (b) Regions are merged into tubes corresponding to singular arcs and tube networks belonging to boundary components of the singularity graph. (c) Interior tubes touching the boundary are connected to the boundary components by chart-zippering on the boundary shell. (d) the algorithm terminates after all charts are merged and all matchings are determined.

each pair of neighboring singular edges is equipped with a tangent continuity constraint of Eq. (16), allowing a series of constrained chart-merging operations to obtain one tubular chart per interior singular arc. Similarly, we perform constrained chart-merging for all boundary sector constraints (18), leading to a closed tubular network for each connected component of the boundary singularities. Pseudocode of these steps is provided in Algorithm 2 and an example visualization of resulting charts is depicted in Fig. 12b.

Step 2: Merging charts at boundary surfaces. In this step we reduce the number of independent charts on the boundary. The key observation is that away from singular edges a consistent frame axis is aligned to the surface normal, reducing the octahedral field topology to the 2D cross-field case. Consequently, we can adapt the 2D zippering approach of [Ray et al. 2008] to consistently merge boundary charts. We process each connected boundary surface independently. First, the boundary triangles of charts containing singularities are used as the root set of a dual spanning forest of all boundary triangles, as shown on the right. For boundary surfaces without singular edges, we choose one arbitrary triangle as the root. Next, for each dual boundary edge d of the spanning tree we merge the charts of both incident triangles by constrained chart-merging for the corresponding interior dual path $\mathcal{D}$.



Since spanning forest edges are never singular, both normals agree in a common chart and we constrain $R_{\mathcal{D}} = I_{3\times3}$. In the final step, we iteratively apply 2D chart-zippering, where the boundary vertex index needs to be taken into account when closing dual cycles. Assume a counter-clockwise (ccw) cycle of dual (non-singular) boundary edges $d_0 \dots d_k$, which surround vertex p and where the matchings on $\mathcal{D}_i$ are known for all i except one. The holonomy of the cycle must be a rotation around the coordinate axis of the normal vector with angle given by the index of the boundary vertex, i.e. $R_{\mathcal{D}_k \dots \mathcal{D}_0} = \mathrm{rot}(\hat{\mathbf{v}}, 2\pi \cdot \mathrm{idx}(p))$, leading to one step of constrained chart-merging. The ccw index $\mathrm{idx}(p)$ of a boundary vertex p is equal to the index of an incident interior singular edge or zero if there is none. In each boundary component without singular edges all boundary tetrahedra will be merged into a single chart. If a boundary surface has k connected sets of singular boundary edges, 2D chart-zippering will terminate with k independent charts, which will only be merged in a subsequent step. Pseudocode of the boundary chart merging is provided in Algorithm 3 and Fig. 12c shows an example resulting in two boundary charts that are separated by the red boundary curve.



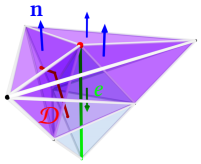
Algorithm 2 MergeSingularArcCharts(X)

```
for all singular edges e do
    X ← MergeAndZipChart(e)                ▷ Singular edge charts
end for
for all tangent continuity constraints T do
    if T relates two different charts then  ▷ Prevent cycles
        X ← ConstrainedChartMerging(T)      ▷ Int. sing. tubes
    end if
end for
for all boundary sector constraints B do
    X ← ConstrainedChartMerging(B)          ▷ Bnd. sing. tubes
end for
return X                                    ▷ Return determined matchings
```

Algorithm 3 MergeBoundaryCharts(X)

```
for all boundary components B do
    Construct dual spanning forest DSF on boundary
    for each dual edge d ∈ DSF do
        X ← ConstrainedChartMerging(D, I_3x3)
    end for
    while ∃ zippable ccw-cycle C enclosing vertex p do
        X ← ConstrainedChartMerging(C, rot(v̂, 2π · idx(p)))
    end while
end for
return X                                    ▷ Return determined matchings
```

Step 3: Merging charts in the volume. In this step we merge interior singular tubes to the boundary charts, including those indirectly linked through corners of the singularity graph. First, for each boundary surface chart we merge all interior singular tubes that already touch the boundary through a vertex. This can be easily done with a constrained chart-merging that matches the $\hat{\mathbf{v}}$ coordinate axis of the outward-pointing surface normal $\mathbf{n}$ of the boundary chart with the negative of the inward-pointing coordinate axis $\hat{\mathbf{u}}$ of the singular edge e, i.e. with the matching $\text{rot}(\hat{\mathbf{w}}, -\pi/2)$. Singular tubes that touch the boundary at both ends are only merged on one side, since we cannot decide the twist of global cycles at this point.



Next, we grow a dual spanning forest rooted at boundary chart tetrahedra to all tetrahedra not part of a boundary or tube chart yet. Unconstrained chart-merging is performed for all dual edges of the spanning forest, i.e. matchings are set to identity. Additionally, iterative chart-zippering is used to resolve all locally decidable matchings. After these two steps, all remaining charts are either boundary charts or isolated singular tube charts that do not touch the boundary.

Further singular tubes can be merged to the boundary charts by considering corner constraints (17). A corner constraint is formed by three dual paths $\mathcal{D}_0$, $\mathcal{D}_1$ and $\mathcal{D}_2$ that express the axis orientation of the three singular edges e_0, e_1 and e_2 of a corner in a common chart. We say that a corner constraint is *decidable*[1] if (i) two singular edges are part of a common boundary chart and all matchings on their $\mathcal{D}_i$ are determined and (ii) the third singular edge is not part of a boundary chart. Assuming w.l.o.g. that e_2 is the non-boundary chart edge, we connect it to the boundary by constrained chart-merging along $\mathcal{D}_2$ with matching constraint

$$R_{\mathcal{D}_2}\hat{\mathbf{u}} = R_{\mathcal{D}_0}\hat{\mathbf{u}} \times R_{\mathcal{D}_1}\hat{\mathbf{u}} \tag{22}$$

directly following from (17). After merging a corner, we perform iterative chart-zippering, which frequently has the positive effect of resolving matchings that render other singular corners decidable. Often, iteratively performing the combination of merging at a locally decidable corner with subsequent chart-zippering is sufficient to resolve all matchings.

If no decidable corner constraint exists, non-local considerations are required to proceed. To bundle spatially distributed constraints, we first partition the triangles of unsolved matchings $T_U \subset T$ into subsets that represent identical degrees of freedom. These are exactly the 2-manifold patches embedded in T_U since fixing the matching of one triangle in such a patch is sufficient to resolve all others by chart-zippering. A patch is *decidable* if it is supported by two independent matching constraints, which uniquely specify the single matching degree of freedom and enable constrained chart-merging.

If neither decidable corners nor patches are available, we explore all potential solutions until a valid one is found. The idea is to search a maximally constrained patch with minimal potential solutions, which are either 4 if it is supported by a matching constraint or 24 otherwise. We randomly pick one candidate solution and continue the algorithm. All other candidates are stored as unfinished branches for later investigation. If the current candidate turns out to be infeasible, we backtrack and search other branches. The same branching strategy is used if the final chart-zippering cannot resolve all matchings. This exhaustive search ensures that whenever a solution exists, it is found. Complete pseudocode of the volumetric chart merging is provided in Algorithm 4.

[1]Another special case of decidable corner arises, if one singular edge belongs to a boundary chart not containing boundary singular edges and the other two belong to isolated int. singular tubes. Any locally valid matchings for $\mathcal{D}_i$ are then globally valid.

Algorithm 4 MergeVolumeChart(X)

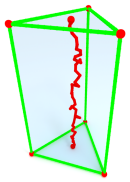
```
for all interior singular tubes T do
    if T incident to boundary then
        X ← ConstrainedChartMerging(T)   ▷ Connect to bnd.
    end if
end for
Grow dual spanning forest DSF from boundary chart tetrahedra
for all dual edges d ∈ DSF do
    X ← UnconstrainedChartMerging(d)
end for
X ← Chart-Zippering(X)                   ▷ Solve locally decidable match.
B ← ∅                                    ▷ Set of unfinished branches
loop
    if ∃ decidable corner C then
        X ← ConstrainedChartMerging(C)   ▷ Eq. (22)
    else
        if ∃ decidable patch P then
            X ← ConstrainedChartMerging(P)
        else
            (i, {m_1 ... m_k}) ← FindBestPatch(X)   ▷ k ∈ {4, 24}
            for j = 2 ... k do
                X_i ← m_j
                B ← B ∪ {X}              ▷ Store candidate for later
            end for
            X_i ← m_1                    ▷ Proceed with first candidate
        end if
    end if
    X ← Chart-Zippering(X)               ▷ Solve locally decidable match.
    if #charts= 1 then
        X ← HandleUnsolvedMatchingsByBranching(X)
        if X satisfies all constraints (15)-(18) then
            return X                     ▷ Valid solution found
        else
            if B ≠ ∅ then
                X ← get next branch of B ▷ Continue search
            else
                return INFEASIBLE
            end if
        end if
    end if
end loop
```

Algorithm Properties. The input to the algorithm is a locally hex-meshable singularity graph that also satisfies the global necessary

condition. The output is a set of matchings of a locally hex-meshable octahedral field if it exists, otherwise the message "INFEASIBLE" is returned. The driving principle of the algorithm is to in each step identify a constraint that can be used to resolve a matching while maintaining feasibility regarding all other constraints. In this sense, it shares similarity with triangular solvers for linear systems of equations, which analogously in each step solve for one variable or choose underdetermined degrees of freedom. Since we cannot guarantee that in each step we can solve for a unique matching, the algorithm additionally includes fallback to exhaustive search of a set of candidate matchings. In this way, termination with a valid solution is guaranteed if one exists. While theoretically necessary, in none of our experiments were multiple branches explored, leading to very fast runtimes in practice. Analyzing the underlying theoretical reason is an interesting direction for future work.

We observed that the singularity graph does not always uniquely specify the octahedral field topology. In particular for higher genus handlebodies, closed singular arcs inside the volume, or cavities, additional constraints are often necessary to uniquely specify field topology. This is not a problem for our algorithm, since by construction it is able to handle underdetermined cases. However, to obtain more control, one could either provide additional constraints or postpone the decision of additional degrees of freedom to the frame generation to, e.g. exploit the degrees of freedom to obtain the smoothest solution.

Relation to Global Necessary Condition. While the global necessary condition Eq. (2) is applied as a filter on the input singular graphs, and helps to detect global inconsistencies, it is not sufficient to guarantee existence of a compatible locally hexable field. Consider the inset example. While the input singularity graph (green lines) satisfies Eq. (2), our algorithm is able to detect that no corresponding locally hexable field exists. Our algorithm produces matchings with extra singularities (red lines) that do not agree with the input singularity graph. While this hints at a method for correcting the singular graph, it is outside the scope of this paper to correct singularity graphs.



5.4 Octahedral Fields with Fixed Matchings and Alignment

After determining the topological matchings and partial alignment information, we still need to construct the frames geometrically to obtain the octahedral field. Regardless of the geometric frame field output, the topology of the frame field is already specified by the topological matchings. However, for hexahedral meshing applications, we aim for the smoothest frame field with prescribed topology. Accordingly, we minimize the Dirichlet energy of a quaternion field q subject to alignment constraints for $q \in B$ resulting from normals and singular edge alignment

$$\begin{aligned} \underset{q}{\text{minimize}} \quad & \int_\Omega |\nabla q|^2 dV \\ \text{subject to} \quad & A_i q_i = 0,\ q_i \in B \\ & |q_i| = 1,\ i = 1 \dots n \end{aligned}$$

with $A_i \in \mathbb{R}^{2\times 4}$ are linear alignment conditions derived below. The $|q_i| = 1$ constraint may be ill-posed in the continuum, an issue addressed using the relaxation below.

Relaxation to eigenvalue problem. The Dirichlet energy is expressed w.r.t. the connection specified by the matchings. Thus, if two orthogonal frames are related through $F_j = F_i R_{j\to i}$, the same relation can be expressed in quaternion form $q_j = q_i \hat{R}_{j\to i}$. While more sophisticated variants are possible, a uniformly weighted discretization of the Dirichlet energy summing squared differences for all dual edges $e^\star_{ij}$ is sufficient for our purposes:

$$\int_\Omega |\nabla q|^2 dV \approx \sum_{e^\star_{ij}} |q_i \hat{R}_{j\to i} - q_j|^2. \tag{23}$$

In the spirit of the globally optimal direction fields of [Knöppel et al. 2013] we relax the unit-norm constraints $|q_i| = 1$ to $\sum_i |q_i|^2 = n$, turning the optimality conditions of our optimization problem into an eigenvalue problem. We add penalty terms $\omega_a |A_i q_i|^2$ to enforce the alignment conditions in the objective function; $\omega_a = 10$ already produces solutions that obey the constraints well.

Axis alignment conditions. The alignment conditions are homogenous linear expressions resulting from the following observation. Assume that we want to align the u-axis of our frame represented by q_i to the direction v. In this case q_i is parametrized by $q_i = q_{\hat{\mathbf{u}}\to v} \cdot q_{\hat{\mathbf{u}}}(\alpha)$, where $q_{\hat{\mathbf{u}}\to v}$ is an arbitrary rotation that maps $\hat{\mathbf{u}}$ to v and

$$q_{\hat{\mathbf{u}}}(\alpha) = \left(\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}, 0, 0\right)^T \tag{24}$$

is a rotation around the first coordinate axis. Since quaternion multiplication is linear in each quaternion we can express the product as $q_i = Q\, q_{\hat{\mathbf{u}}}(\alpha)$, with $Q \in \mathbb{R}^{4\times 4}$ expressing multiplication from the left by $q_{\hat{\mathbf{u}}\to v}$. Hence for normal alignment we end up with the following four constraints on q_i:

$$Q^{-1} q_i = \left(\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}, 0, 0\right)^T$$

It suffices to impose the latter two homogenous constraints because any unit quaternion fulfilling the latter two conditions, i.e. of the form $(q^w, q^x, 0, 0)$, is automatically of the required form (24) for some α. The projection of a given quaternion q to the closest one satisfying a partial alignment condition q_a is done by

$$q_a = Q\,\mathrm{diag}(1,1,0,0)Q^{-1}q$$

with subsequent normalization. We use projection to eliminate small constraint violations resulting from relaxation to penalty terms.

Coping with the double-covering. Since quaternions double-cover SO(3), meaning that q and $-q$ represent the same rotation, there is one additional degree of freedom that we need to address. When specifying the transition functions $\pm\hat{R}_{j\to i}$ in quaternion form, the sign is undefined; hence, we need to ensure that we get the right connection on dual cycles in our mesh respecting the double-covering. To this end, we check and correct all dual edge cycles using an algorithm similar to the chart-zippering of §5.3. We first randomly initialize the signs of $\pm\hat{R}_{j\to i}$ and then whenever a dual cycle is closed, we check whether the first component of the quaternion product along the cycle is positive. If it is not, we invert the sign of

the matching quaternion that closes the cycle. A negative sign in the first quaternion component indicates that the encoded holonomy identifies two different representations in the quaternion double cover. Luckily, the alignment conditions are homogenous and therefore simultaneously valid for $\pm q_i$; this makes them invulnerable to double-cover issues.

6 RESULTS

We evaluate our algorithm by means of several example models of various complexity shown in Figs. 1, 2, 13 and 14. For all examples of Fig. 13 we obtained an initial singularity graph through an octahedral field generated with [Ray et al. 2016]. Based on the necessary local and global conditions of §3 we manually repaired the singularity graph and used the result as input for our algorithm. The resulting matchings and frames then serve as input for [Nieser et al. 2011] to obtain an integer-grid map, which is hex meshed with [Lyon et al. 2016].

The manual correction of a singularity graph was done by iteratively adding, removing and smoothing singular arcs until the singularity graph became locally hexable and satisfied the global necessary condition. If our algorithm reported that the singularity graph was still not globally hexable, we performed further modifications, inspired by the constraints that could not be satisfied. Algorithmic correction is out of the scope of this work but will be an important topic for the future. For the joint model in Fig. 13 we perturbed the input singular graph to demonstrate a result on messy input. Our algorithm only cares about the topology of the input singular graph such that geometrically messy input does not impact the extracted matchings.

The complex examples of Fig. 14 were generated in a similar manner, however, the singularity graphs were imported from hexahedral meshes provided in the supplemental material of [Fang et al. 2016; Fu et al. 2016; Huang et al. 2014; Li et al. 2012]. For all models that we tried, our algorithm generated a valid octahedral field with correct singularity graph as verified by checking the algebraic system for the output field; this empirically verifies correctness of our algorithm. The input complexity ranges from 5k up to 300k tetrahedra for the elephant model. Even for the largest models, generating the matchings only requires a few seconds, while the optimization problem to obtain the frames takes significantly longer, e.g. 40 seconds for the bunny model. For the hand, kitten, knot and rockerarm models, despite having a valid octahedral field [Nieser et al. 2011] was not able to obtain a locally injective IGM, leading to some missing hexahedra or polygonal cells.

Comparison to previous work. Since our method is the first one for generating octahedral fields with prescribed topology, a direct comparison to previous work is impossible. Closest to our method are the singularity correction techniques of [Li et al. 2012] and [Jiang et al. 2014]. While they are automatic, they only perform a small set of local corrections through splitting or collapsing singularities that is not sufficient to obtain hex-meshable singularity graphs for most of our examples. For example, in Fig. 1, global changes and the addition of new singular arcs are necessary. Both previous methods

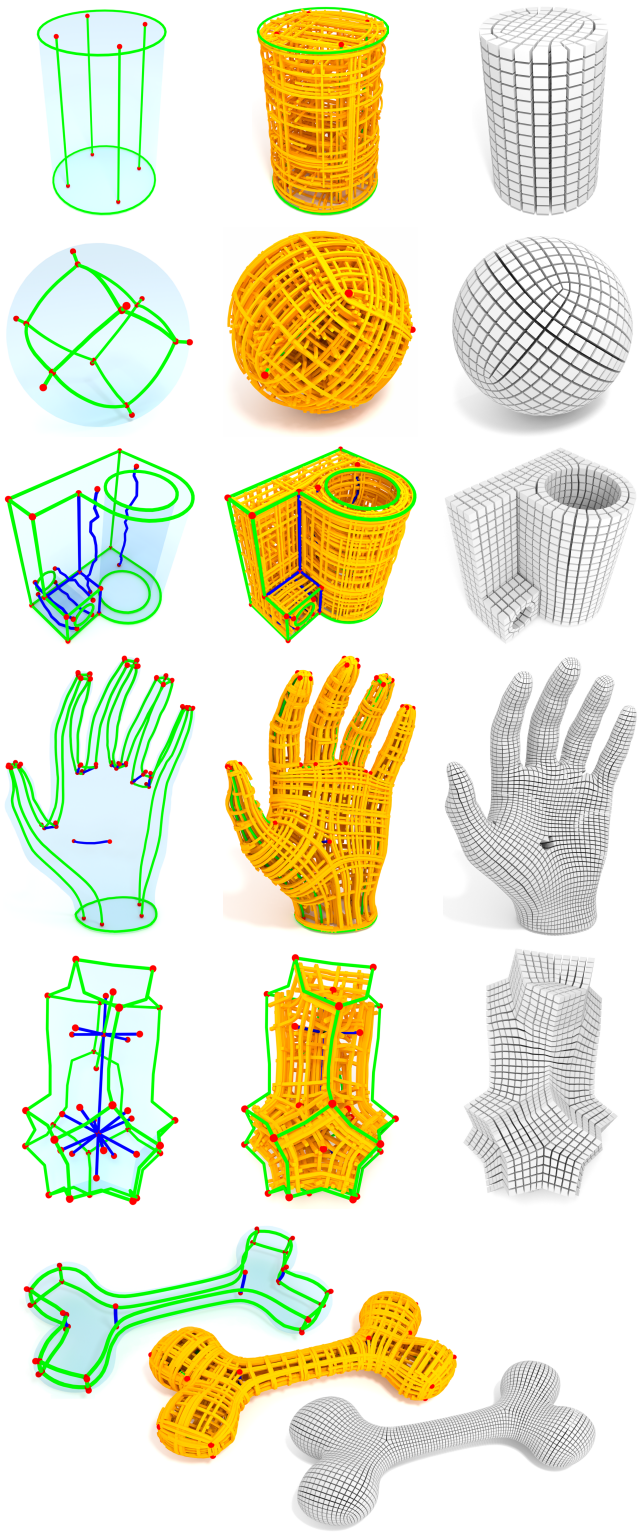


Fig. 13. Several models with manually corrected singularity graph. From left to right: singularity graph, octahedral field and hexahedral mesh.

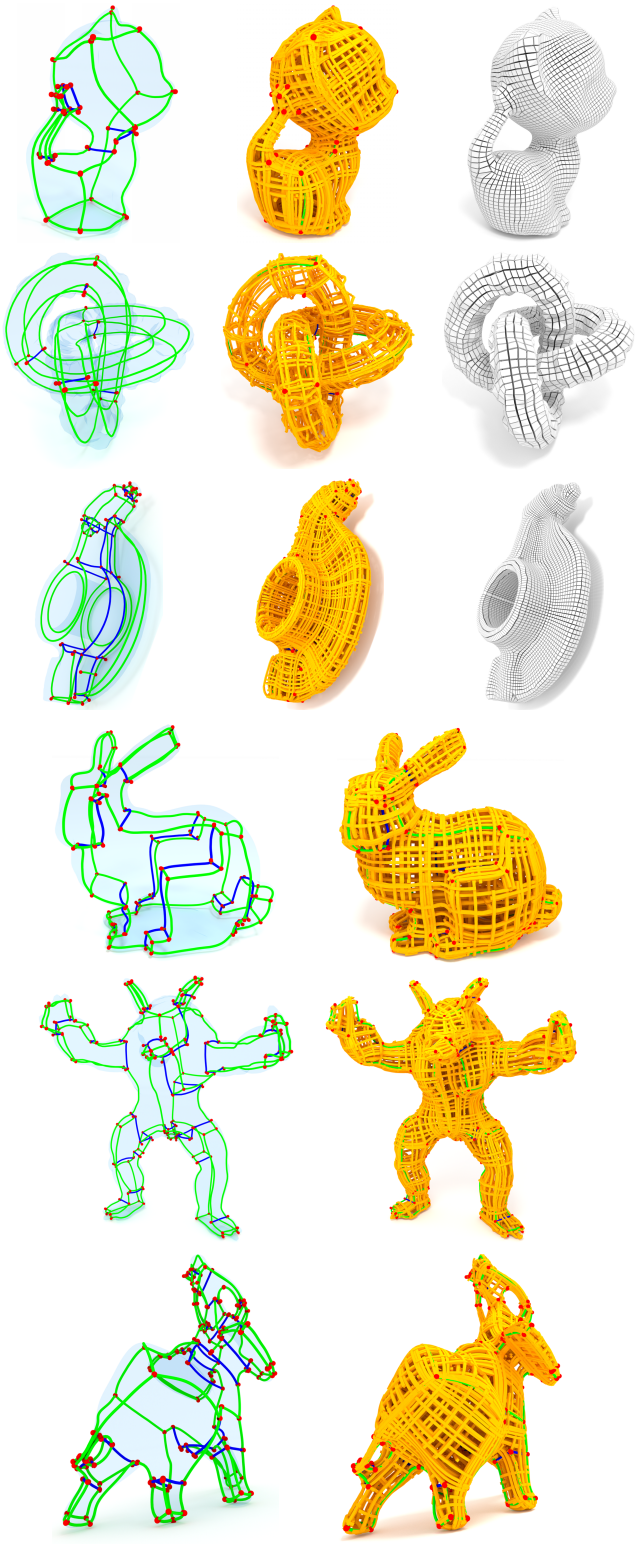


Fig. 14. Complex examples obtained from supplemental material of: [Fang et al. 2016] kitten and knot, [Li et al. 2012] rockerarm, [Fu et al. 2016] bunny and armadillo, and [Huang et al. 2014] elephant.

cannot handle such global changes, which require a re-computation of the octahedral field.

The geometric quality of the hexahedral meshes, typically measured using scaled Jacobians, mostly depends on the pre-processing (design of singularity graph) and post-processing (optimization e.g. via [Livesu et al. 2015]). Consequently, measuring scaled Jacobians is not a meaningful way to evaluate success of our algorithm, since we would mostly measure how much time we spent on tuning the input singularity graph. Our scaled Jacobians are typically on par with the state of the art.

7 CONCLUSIONS AND FUTURE WORK

Many confounding factors make automatic hexahedral meshing extremely difficult; somehow the elegant structure of two-dimensional quad meshing problems does not admit an obvious lifting to the volumetric case. While octahedral field-based meshing appears to be a strong contender for design of practical and efficient algorithms, many questions remain in establishing the fundamentals of this approach. We consider our work to be a serious step toward theoretically-justified, robust field-based hex meshing. By returning to the basics, we clarify the fundamental unknowns in the problem by defining the relevant algebraic system. Along the way we derive new necessary local and global conditions for hex meshability of singular graphs, including a *complete* enumeration of the practically-relevant vertex singularities. Furthermore, our *chart-merging* algorithm robustly recovers topology-constrained octahedral fields that can be provided to existing field-guided meshing techniques.

Several open mathematical and algorithmic challenges remain in the domain of field-guided hex meshing. The holy grail in this domain is a complete (necessary *and* sufficient) characterization of hex meshable singular octahedral field topologies, ideally accompanied by an algorithm that can project a singular graph to its closest meshable counterpart. A continuum theory of octahedral fields posed using differential geometry/topology language rather than relying on an underlying discrete structure may also provide insight. In parallel with these theoretical considerations, other topics for future research are more human-oriented. In particular, while singular topology is critical for understanding the set of hex meshes that can be embedded in a particular volume, it may be the case that a natural user interface for hex meshing should incorporate guidance in a different form, inferring the singular topology automatically behind the scenes. One intriguing direction might be to *learn* a map from volumes to singular graphs informed by a collection of hand-designed hex meshes.

These future improvements aside, we anticipate that our algorithm and theoretical framework will broadly inform the theory and practice of field-based hexahedral meshing. By working in the space of globally hex-meshable singular graphs and correcting octahedral fields to conform to this restricted set, we can circumvent debilitating degeneracies later in the meshing pipeline.

ACKNOWLEDGMENTS

D. Bommes received funding from the German Research Foundation (DFG, grant GSC 111, Aachen Institute for Advanced Study in Computational Engineering Science) and H. Liu from the Chinese Scholarship Council (CSC). J. Solomon acknowledges the generous support of Army Research Office grant W911NF-12-R-0011 ("Smooth Modeling of Flows on Graphs"), from the MIT Research Support Committee ("Structured Optimization for Geometric Problems"), and from the Skoltech–MIT Next Generation Program ("Simulation and Transfer Learning for Deep 3D Geometric Data Analysis"). We would like to thank Jan Möbius for *OpenFlipper*, Martin Heistermann for help with interfacing Blender, Amir Vaxman for inspiring discussions, and the reviewers for their helpful feedback.

REFERENCES

Cecil G. Armstrong, Harold J. Fogg, Christopher M. Tierney, and Trevor T. Robinson. 2015. Common Themes in Multi-block Structured Quad/Hex Mesh Generation. *Procedia Engineering* 124, Supplement C (2015), 70 – 82. 24th Int. Meshing Roundtable.

Tristan Carrier Baudouin, Jean-François Remacle, Emilie Marchandise, François Henrotte, and Christophe Geuzaine. 2014. A frontal approach to hex-dominant mesh generation. *Advanced Modeling and Simulation in Engineering Sciences* 1, 1 (10 Feb 2014), 8.

P.-E. Bernard, J.-F. Remacle, N. Kowalski, and C. Geuzaine. 2016. Frame field smoothness-based approach for hex-dominant meshing. *Computer-Aided Design* 72, Supplement C (2016), 78 – 86. 23rd International Meshing Roundtable.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-grid Maps for Reliable Quad Meshing. *ACM Trans. Graph.* 32, 4, Article 98 (July 2013), 12 pages.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013b. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum* 32, 6 (2013), 51–76.

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

R. Bowen and S. Fisk. 1967. Generations of Triangulations of the Sphere. *Math. Comp.* 21 (1967), 250–252.

Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized Global Parametrization. *ACM Trans. Graph.* 34, 6, Article 192 (Oct. 2015), 12 pages.

Marcel Campen and Denis Zorin. 2017. Similarity Maps and Field-guided T-splines: A Perfect Couple. *ACM Trans. Graph.* 36, 4, Article 91 (July 2017), 16 pages.

Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum* 29, 5 (2010), 1525–1533.

Tamal K. Dey and Sumanta Guha. 1998. Computing Homology Groups of Simplicial Complexes in R3. *J. ACM* 45, 2 (March 1998), 266–287.

Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. 2013. QEx: Robust Quad Mesh Extraction. *ACM Trans. Graph.* 32, 6, Article 168 (Nov. 2013), 168:1–168:10 pages.

Jeff Erickson. 2014. Efficiently Hex-Meshing Things with Topology. *Discrete & Computational Geometry* 52, 3 (Oct. 2014), 427–449.

Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-hex Meshing Using Closed-form Induced Polycube. *ACM Trans. Graph.* 35, 4, Article 124 (July 2016), 124:1–124:9 pages.

Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient Volumetric PolyCube-Map Construction. *Computer Graphics Forum* 35, 7 (2016), 97–106.

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

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

Xifeng Gao, Daniele Panozzo, Wenping Wang, Zhigang Deng, and Guoning Chen. 2017b. Robust Structure Simplification for Hex Re-meshing. *ACM Trans. Graph.* 36, 6, Article 185 (Nov. 2017), 13 pages.

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

Allen Hatcher. 2001. *Algebraic Topology*. Cambridge University Press, Cambridge, UK.

Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. *l*1-Based Construction of Polycube Maps from Complex Shapes. *ACM Trans. Graph.* 33, 3, Article 25 (June 2014), 11 pages.

Jin Huang, Yiying Tong, Hongyu Wei, and Hujun Bao. 2011. Boundary Aligned Smooth 3D Cross-frame Field. *ACM Trans. Graph.* 30, 6, Article 143 (Dec. 2011), 8 pages.

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

Felix Kaelberer, Matthias Nieser, and Konrad Polthier. 2007. QuadCover - Surface Parameterization using Branched Coverings. *Computer Graphics Forum* 26, 3 (2007), 375–384.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally Optimal Direction Fields. *ACM Trans. Graph.* 32, 4, Article 59 (July 2013), 10 pages.

N. Kowalski, F. Ledoux, and P. Frey. 2014. Block-structured Hexahedral Meshes for CAD Models Using 3D Frame Fields. *Procedia Engineering* 82, Supplement C (2014), 59 – 71. 23rd International Meshing Roundtable (IMR23).

N. Kowalski, F. Ledoux, and P. Frey. 2016. Smoothness driven frame field generation for hexahedral meshing. *Computer-Aided Design* 72, Supplement C (2016), 65 – 77. 23rd International Meshing Roundtable.

Michael Kremer, David Bommes, Isaak Lim, and Leif Kobbelt. 2014. *Advanced Automatic Hexahedral Mesh Generation from Surface Quad Meshes.* Springer International Publishing, Cham, 147–164.

Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. 2012. All-hex Meshing Using Singularity-restricted Field. *ACM Trans. Graph.* 31, 6, Article 177 (Nov. 2012), 11 pages.

Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. 2018. Source Code: Singularity-Constrained Octahedral Fields for Hexahedral Meshing. https://graphics.rwth-aachen.de:9000/SCOF/SingularityConstrainedOctahedralFields

Marco Livesu, Alla Sheffer, Nicholas Vining, and Marco Tarini. 2015. Practical Hex-mesh Optimization via Edge-cone Rectification. *ACM Trans. Graph.* 34, 4, Article 141 (July 2015), 11 pages.

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

Loic Marechal. 2009. Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In *Proceedings of the 18th International Meshing Roundtable.* Springer, Berlin, Heidelberg, 65–84.

Tobias Martin, Elaine Cohen, and Robert M. Kirby. 2012. SMI 2012: Mixed-element Volume Completion from NURBS Surfaces. *Comput. Graph.* 36, 5 (Aug. 2012), 7.

Karl Merkley, Corey Ernst, Jason Shepherd, and Michael Borden. 2008. Methods and Applications of Generalized Sheet Insertion for Hexahedral Meshing. In *Proceedings of the 16th International Meshing Roundtable.* Springer Berlin Heidelberg, Berlin, Heidelberg, 233–250.

N. Mishra and D.G. Sarvate. 2008. A Note on Non-Regular Planar Graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing* 66 (2008), 17–31.

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

Jonathan Palacios, Lawrence Roy, Prashant Kumar, Chen-Yuan Hsu, Weikai Chen, Chongyang Ma, Li-Yi Wei, and Eugene Zhang. 2017. Tensor Field Design in Volumes. *ACM Trans. Graph.* 36, 6, Article 188 (Nov. 2017), 15 pages.

Nicolas Ray, Dmitry Sokolov, and Bruno Lévy. 2016. Practical 3D Frame Field Generation. *ACM Trans. Graph.* 35, 6, Article 233 (Nov. 2016), 9 pages.

Nicolas Ray, Dmitry Sokolov, Maxence Reberol, Franck Ledoux, and Bruno Lévy. 2017. *Hexahedral Meshing: Mind the Gap!* Technical Report. INRIA.

Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-symmetry Direction Field Design. *ACM Trans. Graph.* 27, 2, Article 10 (May 2008), 13 pages.

E. F. Schmeichel and S. L. Hakimi. 1977. On Planar Graphical Degree Sequences. *SIAM J. Appl. Math.* 32, 3 (1977), 598–609.

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

Dmitry Sokolov and Nicolas Ray. 2015. *Fixing normal constraints for generation of polycubes.* Technical Report. INRIA.

Dmitry Sokolov, Nicolas Ray, Lionel Untereiner, and Bruno Lévy. 2016. Hexahedral-Dominant Meshing. *ACM Trans. Graph.* 35, 5, Article 157 (June 2016), 23 pages.

Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary Element Octahedral Fields in Volumes. *ACM Trans. Graph.* 36, 3, Article 28 (May 2017), 16 pages.

Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. 2004. PolyCube-Maps. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 853–860.

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. *Computer Graphics Forum* 35, 2 (2016), 545–572.

Ryan Viertel, Matthew L Staten, and Franck Ledoux. 2016. *Analysis of Non-Meshable Automatically Generated Frame Fields.* Technical Report. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States).

Rui Wang, Shuming Gao, Zhihao Zheng, and Jinming Chen. 2016. Frame Field Guided Topological Improvement for Hex Mesh Using Sheet Operations. *Procedia Engineering* 163 (2016), 276 – 288. 25th International Meshing Roundtable.

W. Yu, K. Zhang, and X. Li. 2015. Recent algorithms on automatic hexahedral mesh generation. In *2015 10th International Conference on Computer Science Education (ICCSE).* 697–702.

A LOCAL VERTEX TOPOLOGIES

A.1 Interior Vertices

A triangulation of the sphere with vertices V, edges E, and faces F necessarily obeys Euler's formula: $|V| - |E| + |F| = 2$. As all faces are triangles, we have $3|F| = 2|E|$; and also that $\sum_{v \in V} \deg v = 2|E|$, resulting in the following form:

$$\frac{1}{6} \sum_{v \in V} (6 - \deg v) = 2 \;\Leftrightarrow\; 3i + 2j + k = 12, \tag{25}$$

where i, j, k denote the number of valence 3, 4, 5 vertices in the triangulation. Let the *signature* of a triangulation denote the triplet (i, j, k). As i, j, k are all nonzero integers, there are a finite number of solutions, or possible signatures for such a triangulation, listed below, organized by the total number of vertices $|V| = i + j + k$:

$\|V\| = 4$:	(4,0,0)			
$\|V\| = 5$:	~~(3,1,1)~~	(2,3,0)		
$\|V\| = 6$:	~~(3,0,3)~~	(2,2,2)	~~(1,4,1)~~	(0,6,0)
$\|V\| = 7$:	(0,5,2)	(1,3,3)	~~(2,1,4)~~	
$\|V\| = 8$:	(0,4,4)	~~(1,2,5)~~	(2,0,6)	
$\|V\| = 9$:	(0,3,6)	~~(1,1,7)~~		
$\|V\| = 10$:	(0,2,8)	~~(1,0,9)~~		
$\|V\| = 11$:	~~(0,1,10)~~			
$\|V\| = 12$:	(0,0,12)			

The signatures that have been struck out, correspond to signatures with no valid corresponding triangulation of the sphere. This is a consequence of previous work [Mishra and Sarvate 2008; Schmeichel and Hakimi 1977] which characterizes the possible vertex degrees of planar (and hence sphere) triangulations. The remaining signatures are unique corresponding local singularities, see Fig. 6.

A.2 Boundary Vertices

Enumerating boundary vertex topologies is equivalent to enumerating triangulations of a disc. We enumerate such triangulations by modifying an algorithm for enumerating sphere triangulations [Bowen and Fisk 1967]. Let us first summarize their basic approach. As done in Equation (25), the Euler formula can be rearranged into $\sum_k X_k(6-k) = 12$ where X_k is the number of vertices of degree k. This shows that all triangulations of a sphere must contain some vertices with degree less than 6. For any triangulation with more than 3 vertices, one of these vertices may be removed, resulting in a triangulation with one less vertex. Reversing the vertex removal procedure yields an algorithm for generating all triangulations of a sphere of V vertices from all triangulations of a sphere of $V-1$ vertices. The degree 6 bound means that there are only a finite number of possibilities for each vertex addition. Application of this algorithm and restriction to triangulations with vertex degrees 3,4, and 5 verifies the results of Appendix A.1.

We extend this approach to triangulations of a disc. In this case, rearranging Euler's formula gives $\sum_k X_k(6-k) = 6 + 2N_b$, where N_b is the number of boundary vertices, again forcing existence of some vertices with degree less than 6. The other main difference is that boundary vertex removals are considered in addition to interior vertex removals. These removals are illustrated in Figure 15. Application of the analogous algorithm and restriction to triangulations with vertex degrees 2, 3, 4, and 5 (with degree 2 vertices restricted to the boundary) gives us our result.

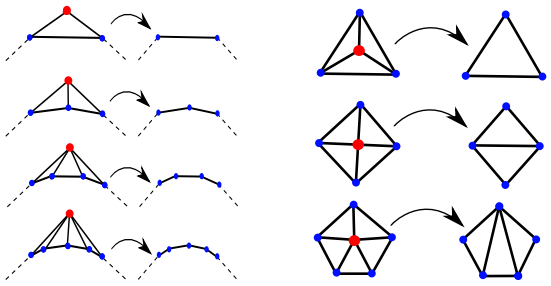


Fig. 15. List of ways to remove a vertex of degree 2, 3, 4, or 5, from a disc triangulation. The red vertex is the one being removed. (Left) Removal of boundary disc vertices. (Right) Removal of interior disc vertices.

B GLOBAL INDEX CONDITION

Condition (2) follows from the following finer condition:

$$\frac{1}{2} \sum_{v \in \partial V} \left(1 - \frac{\mathrm{val}_h(v)}{4}\right) - \sum_{e \in \partial E} \left(\frac{2 - \mathrm{val}_h(e)}{4}\right) + \sum_{v \in \mathring{V}} \left(1 - \frac{\mathrm{val}_h(v)}{8}\right) - \sum_{e \in \mathring{E}} \left(1 - \frac{\mathrm{val}_h(e)}{4}\right) = 0, \tag{26}$$

where $\mathring{V}, \mathring{E}$ denote the interior vertices and edges of the hex mesh. This is argued straightforward by noting that only singular vertices and edges make nonzero contributions to the lefthand side; and for non-closed singular arcs, all but one of the internal component edge contributions is canceled by internal vertex contributions.

To argue for Equation (26), first note two simple equalities:

$$\sum_{v \in V} \mathrm{val}_h(v) = 8|H| \;\;\&\;\; \sum_{e \in E} \mathrm{val}_f(e) = 4|F|\,.$$

With these, consider the standard formula for the Euler characteristic of a 3-dimensional CW-complex:

$$\begin{aligned}\chi(M) &= |V| - |E| + |F| - |H| \\ &= \left(\sum_{v \in V} 1 - \frac{\mathrm{val}_h(v)}{8}\right) - \left(\sum_{e \in E} 1 - \frac{\mathrm{val}_f(e)}{4}\right)\end{aligned}$$

The above terms behave well for interior vertices and edges, but regular boundary vertices and edges still make nonzero contribution, so we focus on these and perform some additional manipulation.

$$\begin{aligned}&\left(\sum_{\partial V} 1 - \frac{\mathrm{val}_h(v)}{8}\right) - \left(\sum_{\partial E} 1 - \frac{\mathrm{val}_f(e)}{4}\right) \\ &\quad= \frac{\partial V}{2} + \frac{1}{2}\left(\sum_{\partial V} 1 - \frac{\mathrm{val}_h(v)}{4}\right) - \frac{\partial E}{4} - \left(\sum_{\partial E} \frac{3 - \mathrm{val}_f(e)}{4}\right) \\ &\quad= \frac{\chi(\partial M)}{2} + \frac{1}{2}\left(\sum_{\partial V} 1 - \frac{\mathrm{val}_h(v)}{4}\right) - \left(\sum_{\partial E} \frac{3 - \mathrm{val}_f(e)}{4}\right)\end{aligned}$$

The last equality follows from Equation (3) for the surface quad mesh. Now, realizing that $\chi(M) = \frac{\chi(\partial M)}{2}$ (see [Dey and Guha 1998] for a discussion), we get Equation (26).