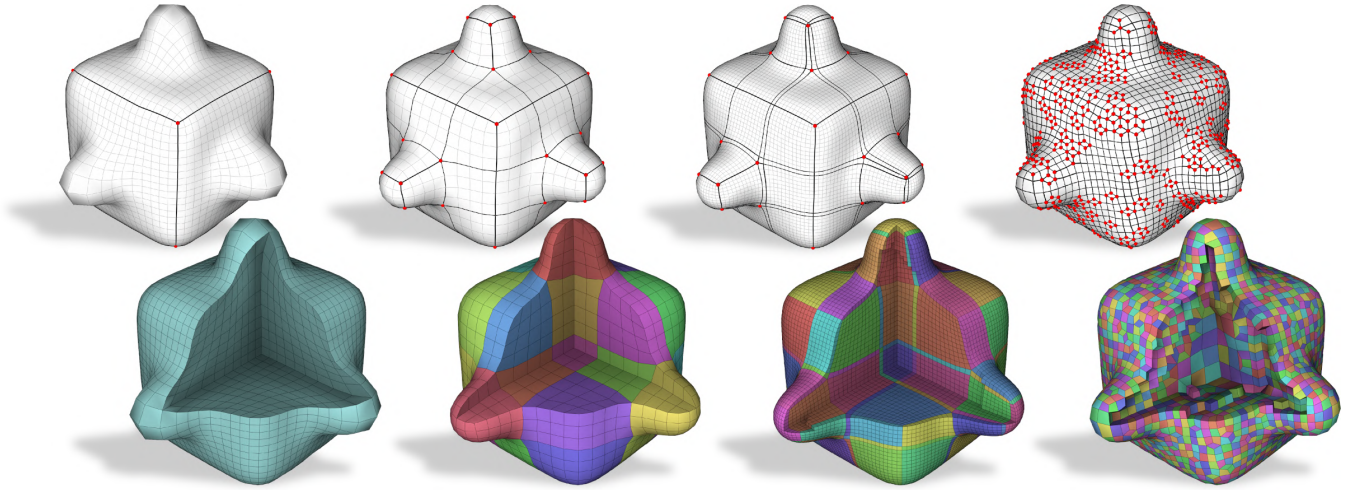
Hex-Mesh Generation and Processing: a Survey

NICO PIETRONI, University of Technology Sydney, Australia
MARCEL CAMPEN, Osnabrück University, Germany
ALLA SHEFFER, University of British Columbia, Canada
GIANMARCO CHERCHI, University of Cagliari, Italy
DAVID BOMMES, University of Bern, Switzerland
XIFENG GAO, Tencent America, USA
RICCARDO SCATENI, University of Cagliari, Italy
FRANCK LEDOUX, CEA, France
JEAN-FRANÇOIS REMACLE, Université catholique de Louvain, Belgium
MARCO LIVESU, CNR IMATI, Italy



Regular	Semi-Regular	Valence Semi-Regular	Irregular
8 irregular vertices	56 irregular vertices	56 irregular vertices	3.7K irregular vertices
1 block	33 blocks	61 blocks	9.7K blocks

Fig. 1. Hex-meshes can be categorized according to their structural regularity, which depends on the amount of irregular edges and vertices present in the mesh (red dots) and on how they are connected to each other (bold lines). Irregularities and their connectivity imply a decomposition of the mesh into regular blocks. The first three meshes were computed with a polycube method [Livesu et al. 2013], the latter with an octree method [Gao et al. 2019].

In this article, we provide a detailed survey of techniques for hexahedral mesh generation. We cover the whole spectrum of alternative approaches to mesh generation, as well as post processing algorithms for connectivity editing and mesh optimization. For each technique, we highlight capabilities and limitations, also pointing out the associated unsolved challenges. Recent relaxed approaches, aiming to generate not pure-hex but hex-dominant meshes, are also discussed. The required background, pertaining to geometrical as well as combinatorial aspects, is introduced along the way.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Mesh models**; **Volumetric models**.

Authors' addresses: Nico Pietroni, nico.pietroni@uts.edu.au, University of Technology Sydney, Australia; Marcel Campen, campen@uos.de, Osnabrück University, Germany; Alla Sheffer, sheffa@cs.ubc.ca, University of British Columbia, Canada; Gianmarco Cherchi, g.cherchi@unica.it, University of Cagliari, Italy; David Bommes, david.bommes@inf.unibe.ch, University of Bern, Switzerland; Xifeng Gao, gxf.xisha@gmail.com, Tencent America, USA; Riccardo Scateni, riccardo@unica.it, University of Cagliari, Italy; Franck Ledoux, franck.ledoux@live.fr, CEA, France; Jean-François Remacle, jean-francois.remacle@uclouvain.be, Université catholique de Louvain, Belgium; Marco Livesu, marco.livesu@gmail.com, CNR IMATI, Italy.

1 INTRODUCTION

Volume meshes explicitly encode both the surface and the interior of an object, thus offering a richer representation than surface meshes. They are primarily used in industrial and biomedical applications, where volume elements are exploited to encode various information, such as structural and material properties, permitting to simulate and precisely estimate the physical behavior of an object, subject to external or internal forces, or the dynamics involving multiple objects interacting in the same environment. Alongside tetrahedra, hexahedral elements are the most prominent solid elements used to represent discrete volumes in computational environments. Meshes entirely or partially made of hexahedra have been used for many years as the computational domain to solve partial differential equations (PDEs) that are relevant for the automobile, naval, aerospace, medical and geological industries to name a few, and are at the core of prominent software tools used by such industries, such as [Altair 2022; ANSYS 2022; CoreForm 2022a; CUBIT 2022; Distene SAS 2022; Tessaels 2022].

In academic research, the algorithmic generation and processing of hexahedral meshes have been studied for more than 30 years now. Despite the huge effort that various scientific and industrial communities have spent so far, the computation of a high-quality hexahedral mesh conforming to (or suitably approximating) a target geometry remains a challenge with various open aspects for which no fully satisfactory solutions have been provided yet. Some of the known methods are extremely robust and scale well on complex geometries; some others produce high-quality meshes; some others are fully automatic. But no known method successfully combines all these properties into a single product. The hex-meshing problem had been so elusive that it was even once termed the "holy grail" of mesh generation [Blacker 2000]. Ever since, many advancements in the field have been made, while major challenges still remain.

In the last decade, the Computer Graphics community has contributed significantly to this field, proposing seminal ideas, theoretical insights, and practical algorithms. In this survey, we wish to summarize this work, also reporting on previous methods developed by other scientific communities. The engineering community has already produced a few surveys on this topic, but they are either no longer up to date [Blacker 2000; Owen 1998; Schneiders 2000; Tautges 2001] or focus just on a particular subset of the available techniques [Armstrong et al. 2015; Sarrate Ramos et al. 2014; Shepherd and Johnson 2008]. We wish to create a comprehensive entry point for researchers and practitioners dealing with hexahedral meshing. We therefore embrace the whole field, revisiting and structuring a vast amount of literature, and covering basic topological (Sec. 2) and geometrical (Sec. 3) concepts, all kinds of approaches to hexahedral mesh generation (Sec. 4), operators to edit mesh connectivity and to perform refinement or coarsening (Sec. 5), mesh optimization and untangling (Sec. 6), visual exploration (Sec. 7), and also addressing the recent trend of methods for hex-dominant meshing (Sec. 4.9). Last but not least, in the final part of the survey, we highlight the current challenges the field is facing and indicate interesting directions for future work (Sec. 8).

2 HEX-MESH STRUCTURE

A hexahedral mesh has structural aspects (concerning the connectivity of mesh elements) and geometric aspects (concerning the elements' shape and their embedding or immersion in space). In this section we focus on the diverse set of structural aspects, and consider geometry in Sec. 3.

2.1 Primal Structure

In terms of connectivity, a hexahedral mesh is a 3-dimensional *cell complex*, $\mathcal{H} = (V, E, F, C)$, consisting of vertices V (0-cells), edges E (1-cells), facets F (2-cells), and cells C (3-cells). The facets F are also often referred to as faces, and the 3-cells C are, given the context, often referred to as hexahedra or hexes. In a *pure hexahedral mesh*, each facet is a topological quadrilateral (i.e., incident to four edges) and each cell is a topological cube (i.e., incident to six such facets). If a relatively small number of facets and cells are of different type (e.g., tetrahedra, prisms, or pyramids) a mesh is called *hexahedral dominant*.

On top of this connectivity structure, a mesh is equipped with a geometric structure, typically an embedding (or immersion) in $\mathbb{R}^3$ (Sec. 3).

Often, instead of assuming fully generic CW or Δ cell complexes [Hatcher 2000], more restricted connectivity definitions are used for practical purposes [Erickson 2013]. A very common one is to assume that each cell has eight *distinct* vertices, i.e., no hexahedron is self-adjacent at a vertex, edge, or facet. Similarly, pairs of edges, facets, or hexes being adjacent via more than one vertex, edge, or facet, respectively, may be ruled out. This simplifies data structures and algorithms; furthermore, many applications assume each hex to be embedded in a geometrically simple way (e.g. straight edges, ruled facets, cf. Sec. 3) which rules out such self-adjacency and multi-adjacency anyway. Sec. 4.2 discusses further application-dependent structural assumptions and requirements.

2.1.1 Singularities. The most *regular* hexahedral mesh is an (infinite) Cartesian grid, where each vertex, edge, and facet is incident to 8, 4, and 2 hexahedra, respectively. General hexahedral meshes contain elements of different local connectivity, which are accordingly called *irregular* or *singular*. Irregular facets simply correspond

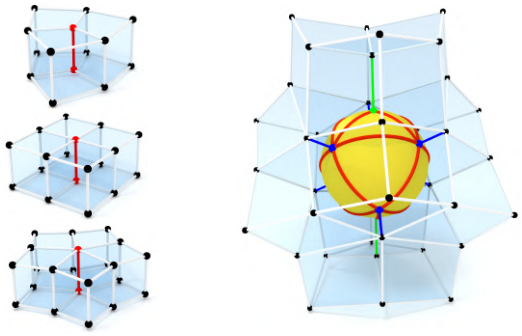


Fig. 2. Left: Singular edges of valence $k = \{3, 4, 5\}$. Right: There is a 1:1-correspondence between configurations of vertices in a hexahedral mesh and triangulations of the sphere. Image from [Liu et al. 2018].

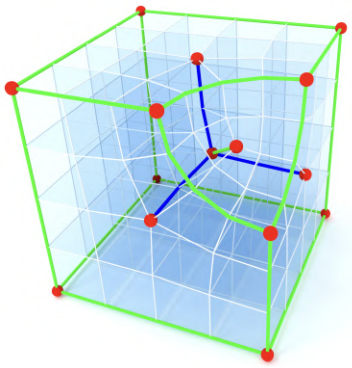


Fig. 3. The singularity graph is formed by interior singular edges of valence $k = 3$ (green) and $k = 5$ (blue) and boundary edges of valence $k = 1$ (green). Chains of edges with homogeneous type connect or terminate at singular vertices (red). Image from [Liu et al. 2018].

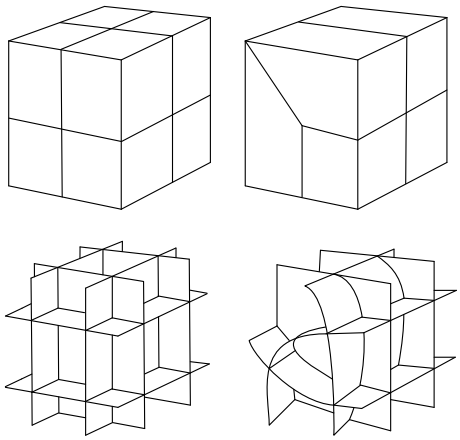


Fig. 4. Two examples of the relation between primal and dual representations. Top: two alternative hexmesh connectivities. Bottom: their corresponding dual cell complexes, formed by arrangements of 2-manifold sheets.

to the *boundary* of the mesh, i.e., all facets that are incident to a single hexahedron.

Since interior facets cannot be irregular and vertex singularities are never isolated [Liu et al. 2018], structurally most interesting is the set of irregular edges, which forms the so-called *singularity graph*.

Singularity Graph. The fundamental building block of the singularity graph is a *singular edge* of valence k, i.e., an interior edge incident to $k \neq 4$ hexahedra, or a boundary edge incident to $k \neq 2$ hexahedra (cf. Fig. 2). While a single integer k is sufficient to characterize the structural type of an edge, the specification of vertex types is more complex. As observed by Nieser et al. [2011] there is a 1:1-correspondence between vertex configurations in a hexahedral mesh, and triangulations of the 2-sphere. This can be understood by observing that the intersection of a cube with a sphere centered at one of it's corners results in a triangular patch (Fig. 2). Hence, different vertex types can be specified by enumerating all triangulations of the sphere. Restricting to (the practically most relevant) edge valences $\{3, 4, 5\}$, it turns out that only 11 different configurations of interior vertex types exist [Liu et al. 2018; Sabin 1997]. Specifically, for an interior vertex it is impossible to be incident to a single singular edge, and in case of two incident singular edges they can only be of identical type. Consequently, the singularity graph is formed by *singular arcs*, which are chains of singular edges with identical type. These singular arcs either terminate at the boundary, or connect to other singular arcs at singular vertices, cf. Fig. 3.

2.2 Dual Structure

In a hexahedral mesh, regardless of its level of structural regularity (Sec. 2.4), each cell has a constant number of 6 facets and each facet has a constant number of 4 edges. Conversely, however, each vertex may have a varying number of incident edges, and each edge a varying number of incident facets.

One may consider the (polyhedral) cell complex that is *dual* to a hexahedral mesh: For each k-cell of the primal mesh there is a $(3 - k)$-cell in 1:1-correspondence in the dual mesh and incidence relationships are adopted. The above regularity of cells and facets in the primal mesh translates into regularity of vertices and edges in the dual. Concretely, except at the mesh's boundary, each dual vertex has a constant number of 6 incident dual edges, and each dual edge has a constant number of 4 incident dual facets. Conversely, dual facets and dual cells are polygons and polyhedra of varying structure. Further details and facts about the dual complex can be found in [Tautges and Knoop 2003].

Depending on the algorithmic context, it may be advantageous to consider the primal or this dual view of a hexahedral mesh. A key reason is the following: While vertices and edges of the primal mesh may be regular or singular, the vertices and edges of the dual mesh are all regular; this is due to the fact that all primal facets are quadrilaterals and all primal cells are hexahedra. The following useful definition of *opposite* edges at a regular vertex and *opposite* facets at a regular edge therefore applies everywhere in the dual mesh.

Opposite Elements. At each regular interior vertex v, there are 6 incident edges. For each edge e_1 of these, there is exactly one edge e_2 among these 6 that does not share a facet with e_1; the edges e_1 and e_2 are called *opposite at* v. At each regular interior edge e, there are 4 incident facets. For each facet f_1 of these, there is exactly one facet f_2 among these 4 that does not share a cell with f_1; the facets f_1 and f_2 are called *opposite at* e. In the primal setting, this concept of opposite edge is relevant for algorithms that trace internal arcs in the mesh, e.g., connecting pairs of singular vertices. Similarly, opposite facets are useful to flood internal facet sheets bounded by singular arcs, e.g., to perform a coarse block decomposition of a given mesh (Sec. 2.3).

In the dual setting, this opposite relation can be used to define the following:

Sheets. Consider the transitive closure of the dual facets' opposite-relation. Its equivalence classes are called *sheets* (also referred to as

twist planes [Murdoch et al. 1997] or (pseudo-)hyperplanes [Tautges and Knoop 2003]). These sheets are 2-manifold surfaces (with boundary, possibly self-intersecting) formed by dual facets.

Chords. Analogously, the equivalence classes of the dual edges' opposite-relation's transitive closure are referred to as *chords* [Borden et al. 2002a; Murdoch et al. 1997] (or *polychords* [Daniels et al. 2008]).

The combinatorial "continuity" of opposite dual facets across dual edges has inspired the early name *spatial twist continuum* for this dual sheet based perspective.

It follows that the entire dual complex can be viewed as an arrangement of intersecting manifold surfaces (sheets): dual vertices are formed by three intersecting sheets, chords are formed by two intersecting sheets and split into dual edges by transversely crossing sheets, sheets are split into dual facets by crossing sheets, and dual cells are the spatial compartments enclosed by sheets. Conceptually, a sheet corresponds to one *layer* of hexahedra in the primal mesh; how this layer is composed of individual hexahedra, however, is not defined by this sheet itself but by sheets that cross this sheet transversely. Fig. 4 illustrates this primal-dual relationship.

All this is in close analogy to dual complexes in the case of quadrilateral meshes. These can be viewed as arrangements of intersecting 1-manifolds [Campen et al. 2012; Campen and Kobbelt 2014]. Quite differently, though, sheets can be topologically quite complex, they may have arbitrary genus and an arbitrary number of boundary loops, whereas in the quadrilateral mesh case each 1-manifold may only be either a closed loop curve, or an open-ended curve starting and ending at the mesh boundary.

2.3 Block Structure

Each hexahedral mesh can be decomposed into disjoint blocks, where each block is a regular grid of hexahedra. Conversely, the mesh can be viewed as disjoint union of such blocks. As an extreme example, each hexahedron could be considered an individual block (of size 1×1×1). We can distinguish conforming and non-conforming block decompositions: a block decomposition is conforming iff each side of each block coincides with one other block side (except at the mesh boundary).

Of particular practical relevance are decompositions that are conforming, and among these those that are coarse, i.e., that consist of a small number of blocks. Meshes rarely have a unique conforming block decomposition. The *coarsest* conforming block decomposition is sometimes referred to as the mesh's *base complex* [Bommes et al. 2011; Brückler et al. 2022b; Gao et al. 2015; Razafindrazaka and Polthier 2017].

It is worth pointing out that the term base complex is sometimes used with alternative meanings [Dong et al. 2006; Eck and Hoppe 1996; Hormann et al. 2008; Livesu et al. 2013], for instance to refer to a coarse cell complex that is used as a domain for (cross)-parametrization. Note that this is not entirely unrelated though: a common use case of these parametrizations is structured remeshing; the resulting meshes typically exhibit a block structure induced by the underlying domain complex.

The base complex has the following defining property: a facet is part of a block side if and only if it is transitively incident to a singular edge via opposite facets (as defined in Sec. 2.2). This suggests a simple way to extract the base complex of a given hexahedral mesh: starting from all facets incident to any singular edge, iteratively expanding through opposite facets across regular edges until termination [Brückler et al. 2022b; Gao et al. 2015]. Due to the practical relevance of semi-regular hexahedral meshes (Sec. 2.4) mesh generation algorithms that take the coarseness of the implied base complex into account are of particular interest.

2.4 Structural Regularity

Similarly to quad-meshes [Bommes et al. 2013b], hex-meshes can be roughly organized into four classes depending on the degree of regularity of their topological structure. The concept of mesh regularity is closely related with the relative amount of irregular vertices present in the mesh and with how these vertices are connected to each other.

- **regular** (or structured) meshes have the topology of a gridded cube (Fig. 1 left). These meshes are extremely convenient for storing and processing because of their simple connectivity: each internal vertex has exactly the same number of neighbors, with a consistent ordering. This allows for efficient storage and optimal query time, and also makes the computation of local quantities (e.g., finite differences) straightforward. There are, however, severe limits in the class of shapes they can represent adequately: mapping the grid into an object containing long protrusions or deep cavities likely results in a mesh with little practical utility due to the poor shape of its distorted elements. Moreover, the rigid global structure does not allow for localized refinement: if more vertices are necessary around a specific area, the entire grid must be refined in order to maintain the regular structure;

- **semi-regular** (or semi-structured, also block-structured) hex-meshes are obtained by gluing in a conforming way several regular grids (also called *blocks*). All vertices that are internal to a block are regular; only vertices that lie at the edges or corners of a block may be irregular. Semi-regular meshes represent a particularly important class in terms of applications, and today are often the result of a manual or semi-manual meshing process. Differently from regular meshes they allow for higher flexibility and can be used to represent shapes of arbitrary complexity. At the same time, they contain a limited amount of irregular vertices, connected to each other so as to define a coarse block layout (Fig. 1, middle left) which can be exploited by dedicated data structures for cheaper storage and fast querying [Tautges 2004], and is also useful in a variety of applications that exploit the tensor product structure of its elements (e.g., IGA [Hughes et al. 2005]);

- **valence semi-regular** meshes also contain a limited amount of irregular vertices, but they are not connected in a way that induces a coarse block decomposition into few regular blocks (Fig. 1, middle right). Meshes of this kind are often produced by modern hex-meshing algorithms such as frame field based methods, which introduce few singularities, thereby creating

meshes with many regular regions, but do not specifically address their connectivity pattern;

- **irregular** (or unstructured) hex-meshes contain a large fraction of irregular vertices (Fig. 1, right). Meshes of this kind are, for instance, produced via voxelization or other grid-based methods: portions of the object that do not align with the ambient Cartesian grid exhibit a typical staircase effect, triggering a proliferation of irregular vertices on the surface. Meshes of this type are most limited in terms of their practical utility and benefits relative to, e.g., tetrahedral meshes.

As for the quad-mesh case [Bommes et al. 2013b] the boundaries between semi-regular, valence semi-regular, and irregular meshes are blurred. Nevertheless, from an applicative perspective there is a substantial difference between these three classes and there exists a variety of structure enhancement algorithms that are specifically designed to improve mesh regularity (Sec. 5.5). The whole taxonomy can be understood in terms of the ratio between the number of irregular vertices and the total amount mesh vertices (r_V), and the ratio between the number of blocks and the total number of mesh elements (r_B). If both r_V and r_B are high, the mesh is irregular; if r_V is low but r_B is high, the mesh is valence semi regular; if both r_V and r_B are low the mesh is semi-regular. Finally, if the number of blocks is exactly 1, the mesh is regular. Providing actual thresholds to precisely define what *high* and *low* mean, is an application dependent matter.

2.5 Integer-Grid Maps

Integer-grid maps (IGM) are a class of maps which by construction induce (at least) valence semi-regular meshes. The central idea, as illustrated in Fig. 5, is to embed an n-dimensional shape into an n-dimensional voxel grid such that the inverse map deforms the set of covered voxels into a shape-aligned hexahedral mesh. So far, integer-grid maps have been studied for 2-manifolds to generate quadrilateral meshes [Bommes et al. 2013a; Kälberer et al. 2007] and for 3-manifolds to generate hexahedral meshes [Liu et al. 2018; Nieser et al. 2011]. Similar to the parametrization of a general manifold, an integer-grid map can be decomposed into multiple charts. However, in order to guarantee that the inversely mapped voxels stitch conformingly, it is necessary to require specific transition functions that preserve the voxel grid. Assuming that the vertices of the voxel grid are given by integer coordinates $\mathbb{Z}^n$, the grid-preserving transition functions are exactly (i) integer translations and (ii) symmetry transformations of an n-cube. Such transition functions are essential to generate meshes with interior singularities, as for instance, the singular vertex (red) in Fig. 5.

Mathematically, a map requires three properties to be an *integer-grid map*: (i) grid-preserving transition functions, (ii) local injectivity, and (iii) singularities and boundaries mapping to integer-grid entities. A thorough definition can be found in [Liu et al. 2018].

Integer-grid maps are sufficiently expressive to describe all potential hexahedral meshes. We can trivially generate a chart for each hexahedron that maps it to the voxel $[0, 1]^3$. In this sense, integer-grid maps can be seen as an alternative representation of

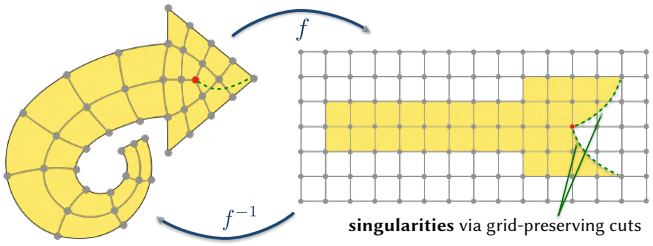


Fig. 5. An integer-grid map f deforms the shape (left) such that its boundary aligns with a Cartesian grid of integer isolines (right). Consequently, its inverse f^{-1} deforms the covered grid cells into a structure-aligned mesh. Grid-preserving cuts (dashed green) enable irregular vertices (red) in the mesh.

hexahedral meshes that has proven highly valuable for designing powerful generation algorithms.

Reformulating the hexahedral mesh generation task as a map optimization problem offers many advantages. First of all, the optimization of low-distortion maps is a well-studied topic with a rich body of theory and algorithms that serve as a strong foundation. Moreover, the map optimization perspective enables multiple geometrically motivated continuous relaxations that are crucial for efficiently finding good approximate solutions of the hard underlying mixed-integer problem, e.g. frame-fields to find suitable singularities (cf. Sec. 4.8), or seamless maps to estimate the required integer translations (cf. [Brückler et al. 2022a; Nieser et al. 2011]).

While a naive direct optimization formulation for a hexahedral mesh needs to explicitly encode and deal with the full set of (inherently discrete) elements and their connectivity, most of that becomes implicit in the map formulation, enabling not only straightforward continuous relaxations but moreover a reduced set of discrete variables. A simple but instructive example consists of a regular block covering $n \times m \times o$ voxels in the IGM image. Stretching the image along the first coordinate axes corresponds to a continuous relaxation of the discrete action of changing the integer dimension n. Note that from the map perspective, the block is indeed fully characterized by only three integers n, m, and o, while a direct mesh optimization would need to deal with $(n + 1) \times (m + 1) \times (o + 1)$ (discrete) vertices and their nontrivially-constrained connectivity. The number of integer degrees of freedom of a general integer-grid map is proportional to the number of singularities and topological handles. Consequently, in case of pre-determined singularities the resulting discrete search space is comparatively small since typically highly regular meshes with only few singularities are desired.

Optimizing for a low-distortion map has two positive effects, (i) it directly promotes well-shaped elements of high quality in the output hex-mesh, and (ii) it demotes the occurrence of spurious singularities.

A conceptual overview of interpreting mesh optimization as map optimization is shown in Fig. 6. The advantages related to superior continuous relaxations and compact discrete search spaces explain the popularity and success of integer-grid map based approaches in the automatic generation of structured meshes.

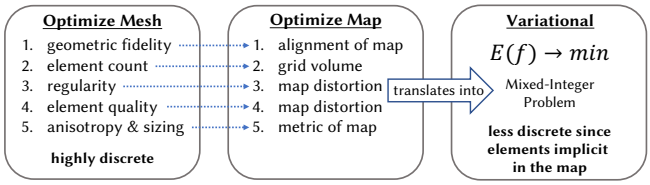


Fig. 6. Integer-grid map optimization algorithms turn the highly discrete mesh optimization task into a more tractable mixed-integer map optimization. All mesh quality criteria are directly related to properties of the map, as indicated by blue dashed arrows.

A special case of integer-grid maps with no interior singularities are *polycube maps*, which are discussed in more detail in Sec. 4.7. The generation of a polycube map, therefore, can be viewed as deforming the input shape such that its surface aligns with a blocky surface from the voxel grid – a polycube. While this deformation can be pictured as a continuous process, the underlying optimization problem is nonetheless of mixed-integer type [Mandad et al. 2022]. The discrete degrees of freedom include the choice, per piece of the shape's surface, which of the six oriented coordinate axes $\{\pm x, \pm y, \pm z\}$ it shall align to.

Frame-field based methods, which are discussed in more detail in Sec. 4.8, target the generation of an integer-grid map (whether general or restricted to polycubes) in two stages. From a high-level perspective, the first stage estimates the rotational part of the Jacobian of an integer-grid map, i.e., a frame-field, while the second stage constructs the map by inheriting the frame-field singularities. The decomposition is beneficial because the first stage can be formulated in a representation that more easily deals with the discrete symmetry of the hexahedron.

3 HEX-MESH GEOMETRY

Besides its combinatorial and topological structure (Sec. 2), a hexahedral mesh's geometry, i.e., its embedding or immersion, typically in $\mathbb{R}^3$, plays an essential role in most applications.

This concerns the question of geometric fidelity (to what extent the mesh conforms to the target shape) and the question of element quality. This latter question is concerned with the *shape* of a mesh's individual hexahedra or the distortion of *maps* defining these hexahedra as deformations of an ideal (*reference* or *master*) element.

Depending on the application context, various geometric requirements may be in place: the mesh may be required to conform to a given boundary mesh or to interpolate it within some prescribed tolerance; facets may be required to be planar or to be convex; the above maps may be required to be locally injective or even to have bounded distortion in some particular sense. In the context of mesh generation (Sec. 4) the concrete requirements can have a significant influence on the hardness of the meshing problem. Many methods so far are unable to provide strict guarantees regarding such requirements, especially when they are asked for in combination.

Also the relevant notion of element quality, and the effect of low or high-quality elements, are application dependent. In the context of simulations by means of finite element methods (FEM), element quality can have a crucial impact on error estimates and convergence rates, thus simulation speed and accuracy [Ciarlet 2002; Zlámal 1968], and relevant quality measures depend on the type of simulation. In Sec. 4.2 these varying requirements are discussed further.

The Trilinear Element. The geometry or embedding of hexahedral meshes is often represented by means of coordinates assigned to their vertices. This alone is sufficient only for simple applications. More commonly, the geometry of edges, faces, and cells has to be defined as well. A particularly simple (and common) scenario is the assumption of *trilinear* elements (linear edges, bilinear faces, trilinear cells), as this does not require the specification of any further information—all other mesh elements' geometric embedding in $\mathbb{R}^3$ are derived from the vertex positions via multilinear interpolation. Precisely, a hexahedron's embedding (with vertex positions p_{ijk}) is defined via a *geometric map* $\tau : [0,1]^3 \rightarrow \mathbb{R}^3$ (also called *isoparametric map*) as follows:

$$\tau(u,v,w) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} p_{ijk} B_k(w) B_j(v) B_i(u),$$

where

$$B_0(t) = 1 - t \text{ and } B_1(t) = t.$$

The hexahedral element effectively is the image of an ideal cube $[0,1]^3$ under this map. Note that the edges are straight line segments under this map; the faces are ruled surfaces (planar iff the four corner vertices are coplanar). More generally, this definition can be extended to higher-order elements using higher-order basis functions B_i^n (e.g., Bernstein polynomials of degree n, giving rise to tensor-product Bézier elements [Prautzsch et al. 2002]). In these higher-order cases, additional control points (besides the vertex points) come into play as coefficients for a higher number of basis functions.

The assessment of these elements' quality (or even just validity) is an application-dependent matter. In some cases it may be just the *shape* of the region $\tau([0,1]^3) \subset \mathbb{R}^3$ that is of relevance, in others its concrete *parametrization*, given by the map τ, is crucial.

3.1 Geometric Map

A particularly common measure of quality is the determinant of the geometric map's Jacobian J_τ. It quantifies to what extent the hexahedron, defined through τ, deviates (in terms of volume distortion) from the cube $[0,1]^3$. Note that $\det J_\tau$ depends on parameters (u,v,w). Due to this dependence, the quality of an element (in contrast to the quality at a particular point) rather needs to be assessed by the extremal value $\min_{[0,1]^3} \det J_\tau$.

Note that $\det J$, while measuring volume distortion, is blind to angle distortion; it cannot distinguish sheared cubes from cubes. Additional angle-aware measures are thus often taken into account (Sec. 3.2).

3.1.1 Element Validity. If $\min_{[0,1]^3} \det J_\tau \leq 0$, the geometric map τ is non-injective and the implied element is said to be irregular. Sometimes a distinction is made between degeneration ($\det J_\tau = 0$) and inversion or fold-over ($\det J_\tau < 0$).

In the context of the finite element method, irregular elements must be considered invalid [Knupp 2000; Mitchell et al. 1971]; with such elements, depending on the concrete setting, one may yield "inaccurate solutions or no solutions at all" [Barrett 1996], solutions are "invalidated" [Roca et al. 2012], or "calculations cannot be continued" [Salagame and Belegundu 1994]. Due to this crucial importance, specialized *untangling* methods for the purpose of irregular element removal in hexahedral meshes have been proposed (Sec. 6), that attempt to achieve $\min_{[0,1]^3} \det J_\tau > 0$.

3.1.2 Computation. The evaluation of $\det J_\tau$ at a concrete parameter point (u, v, w) is quite easy. For the computation of the extrema $\min/\max_{[0,1]^3} \det J_\tau$, however, there is no closed-form expression. As this is particularly relevant to *certify* regularity, simply probing at a number of well-distributed parameter points is a risky approach.

Determinant Bounds. Like τ, the Jacobian determinant is a polynomial in (u, v, w). It can thus be expressed in the Bernstein basis as well:

$$\det J_\tau(u, v, w) = \sum_{ijk} d_{ijk} B_k(w) B_j(v) B_i(u).$$

Due to this basis' implied convex hull property (due to $0 \leq B_i(t) \leq 1$ for $t \in [0, 1]$) the function value is bounded from below by the smallest coefficient $\min_{ijk} d_{ijk}$ and from above by the largest coefficient $\max_{ijk} d_{ijk}$. The coefficients d_{ijk} are easily computed from the vertex points p_{ijk}. For the particular case of trilinear hexahedral elements, this is discussed in [Johnen et al. 2017]. The same principle applies to higher-order elements as well as to simplicial (rather than tensor-product) elements [Dey 1999; Gravesen et al. 2014; Johnen et al. 2013; Luo et al. 2002; Mandad and Campen 2020].

These bounds can be quite loose. They can, however, be tightened arbitrarily by re-expressing $\det J_\tau$ piecewise over subdomains of $[0, 1]^3$ [Hernandez-Mederos et al. 2006]. This is accomplished (via affine reparametrization) using Bézier subdivision [Prautzsch et al. 2002]. Under repeated subdivision, the coefficients (and thus the derived bounds) converge to the actual function value [Leroy 2008; Prautzsch and Kobbelt 1994].

For use cases where precise knowledge of the Jacobian determinant's value range is not relevant but only injectivity is to be certified, simpler (possibly loose) conservative tests can be employed [Zhang 2005]. Various even simpler hypothetical tests (trying to derive bounds from determinant values at vertices or along edges) were shown to be false [Knupp 1990; Zhang 2005].

Relaxation. Through sum-of-squares (SOS) relaxation, the non-convex problem of finding the Jacobian determinant polynomial's global minimum (i.e., $\min_{[0,1]^3} \det J_\tau$) can be replaced by a convex problem [Marschner et al. 2020]. If a sufficiently high degree is chosen for the formulation of this replacement problem, the global minima coincide. A sufficient degree was determined empirically; a formal guarantee is outstanding.

3.2 Shape Quality

Besides metrics based on the pointwise assessment of the geometric map, there exist a variety of metrics based simply on the vertex positions that have been proposed in the literature to assess the quality of hexahedral elements or have been exploited in specific applications. The documentation of the Verdict library [Stimpson et al. 2007] – a de facto standard for finite element mesh quality assessment – exhaustively reports per-hex metrics, as well as associated bounds and commonly acceptable ranges. We succinctly report these metrics in Tab. 1. For more details on how each metric is formulated, we point the reader directly to the original source. It must be noted, though, that the question whether an element is good or at least acceptable can be highly application dependent; in FEM, for instance, elements far from being cube-shaped (in particular anisotropically stretched elements) can be ideal – if they are aligned suitably, in a PDE-guided or even solution-adaptive manner [Knupp 2007].

Table 1. List of alternative metrics for hexahedral elements, from the Verdict library [Stimpson et al. 2007]. Normal ranges are intended for elements not having degeneracies.

Metric	Overall range	Acceptable range	Value for unit cube
Diagonal	$[0, 1]$	$[0.65, 1]$	1
Dimension	$[0, \infty)$	app. dep.	1
Distortion	$[0, 1]$	$[0.5, 1]$	1
Edge Ratio	$[1, \infty)$	—	1
Jacobian	$(-\infty, \infty)$	$[0, \infty)$	1
Max. Edge Ratio	$[1, \infty)$	$[1, 1.3]$	1
Max. Asp. Frobenius	$[1, \infty)$	$[1, 3]$	1
Mean Asp. Frobenius	$[1, \infty)$	$[1, 3]$	1
Oddy	$[0, \infty)$	$[0, 0.5]$	0
Relative Size Squared	$[0, 1]$	$[0.5, 1]$	—
Scaled Jacobian	$[-1, 1]$	$[0.5, 1]$	1
Shape	$[0, 1]$	$[0.3, 1]$	1
Shape and Size	$[0, 1]$	$[0.2, 1]$	—
Shear	$[0, 1]$	$[0.3, 1]$	1
Shear and Size	$[0, 1]$	$[0.2, 1]$	—
Skew	$[0, 1]$	$[0, 0.5]$	0
Stretch	$[0, 1]$	$[0.25, 1]$	1
Taper	$[0, \infty)$	$[0, 0.5]$	0
Volume (signed)	$(-\infty, \infty)$	$[0, \infty)$	1

4 HEX-MESH GENERATION

In this section, we survey the variety of mesh generation techniques present in the literature to date. We firstly provide a general introduction about input and output requirements. Then, algorithms will be organized according to the meshing paradigm they implement. The generation of hybrid, in particular hex-dominant, meshes containing spurious non-hexahedral elements is also discussed (Sec. 4.9). Finally, Tab. 2 summarizes the main properties of each class of hex-meshing algorithms reported in this survey.

4.1 Input

Input data can be either a surface or a volume mesh describing the target geometry. Methods that take a surface mesh or other surface description and produce a conforming hexahedralization are often

called *direct* [Shepherd and Johnson 2008], as opposed to *indirect* methods, which typically operate on a supporting tetrahedral mesh and produce hexahedra by modifying this mesh (through splitting, clustering, etc.) or by computing some volumetric mapping encoded on the vertices of this supporting mesh.

The most trivial form of indirect hex-meshing consists of splitting each tetrahedron into four hexahedra via midpoint refinement [Li et al. 1995]. This technique is trivial to implement and always guarantees a correct result. However, it produces an unstructured mesh with an overly dense singular structure, also containing four times more elements than the input mesh. Therefore, this approach is unsuitable for real applications. As will come clear in the remainder of this section, indirect hex-meshing has evolved significantly since these early days and now comprises highly advanced tools to convert a tet-mesh into a much coarser hex-mesh with cleaner singularity structure. Notably, indirect approaches that cluster tetrahedra to form hexahedra are quite predominant in hex-dominant meshing (Sec. 4.9).

Most of the techniques discussed in this section make assumptions on the topology and geometry of the input mesh and are not able to operate on meshes containing topological (e.g., open boundaries, holes, or non-manifold elements) or geometric (e.g., intersecting or degenerate elements) defects. Methods that operate on a supporting tetrahedral mesh may leverage robust tetrahedralization techniques such as [Diazzi and Attene 2021; Hu et al. 2020, 2018]. Methods that operate on surface meshes can sanitize their inputs with known robust surface processing algorithms, such as [Attene 2010; Attene et al. 2013; Cherchi et al. 2020; Zhou et al. 2016].

In addition to the target geometry, algorithms may optionally take as input a variety of other desiderata, such as target edge lengths or density fields to control local element size and anisotropy, or a list of features that the output mesh should conform to. Typical features are geometric curves on the outer surface (i.e., sharp creases), but there may also be additional ones – both internal and external – such as separation membranes between different materials, or other forms of semantic attributes. Finally, methods based on guiding fields (see Sec. 4.8) may also take as input some additional parameters that control the field generation, or may even assume the whole guiding field as an input by itself.

4.2 Output

Output meshes must satisfy a variety of requirements, some of them strictly, some others loosely. In the following we list the most important topological and geometric requirements, also connecting them with specific applications that demand their fulfillment. The main **topological** desiderata are:

- **element type:** methods that strive for pure hexahedral meshing must ensure that all their cells are topological cuboids made of 8 vertices, 12 edges, and 6 quadrilateral faces. This requirement is loosened for hex-dominant methods, where spurious non-hex elements may be present in the output mesh. This topological freedom is not unlimited, and may be bounded by the specific application. In fact, methods for the numerical solution of PDEs often require non-hex elements to belong to a restricted class of polyhedra. For example, the Poly-Spline Finite Element Method [Schneider et al. 2019] demands that all mesh elements (non-hexahedra included) have quadrilateral faces, and enforces this property through mesh subdivision if the input mesh does not fulfill this requirement. Similar restrictions are also imposed by alternative methods;

- **local structure:** topological limitations may apply not only at a local (per element) level, but also involve clusters of adjacent cells. For instance, the Poly-Spline Finite Element Method [Schneider et al. 2019] requires that two non-hex cells are not face-, edge-, or vertex-adjacent, and also that non-hex cells are not exposed on the boundary. More generally, many methods that employ higher order basis functions can handle just a few local configurations, and put constraints on the local mesh patterns. This holds for both hex and hex-dominant meshes. For example, the blended spline method for unstructured hexahedral meshes proposed in [Wei et al. 2018] embraces only a small fraction of the possible singularities that are created by the meshing methods surveyed in this section. To this end, the intricate mesh connectivity generated by grid-based methods can be extremely challenging [Livesu et al. 2021];

- **global structure:** depending on how the singular elements align, the mesh may or may not have a coarse block structure (Sec. 2.4). While basic numerical schemes like the Finite Element Method operate at a local (per element) level and may not exploit this property, block-structured meshes may be highly important for methods that employ tensor product constructions per block, for multi-grid solvers that rely on a hierarchy of nested meshes, and also for mesh compression [Tautges 2004];

- **conformity:** some hex-dominant methods restrict their output to a narrow class of alternative polyhedra (e.g., permitting only tetrahedra and hexahedra). On the positive side, this restricts the alternative types of cells that applications must handle. On the negative side, the resulting meshes may be *non-conforming*, meaning that structural discontinuities arise between elements that are geometrically but not topologically adjacent (due to T-junctions). Topological continuity can be restored using special *connector* elements of zero volume. Nevertheless, the resulting meshes (with or without connectors) are not supported by all numerical solvers, and dedicated numerical schemes (e.g., Discontinuous Galerkin [Chan et al. 2016]) must be used.

From the geometric point of view, the output meshes should faithfully represent the target shape, preserve its prescribed features (if any), and be composed of well-shaped elements. More precisely, the main **geometric** desiderata are:

- **fidelity:** geometric fidelity is achieved by construction by methods that generate hex meshes conforming to an input quadrilateral surface mesh. Conversely, many other methods typically deviate from the target geometry and may only produce a geometric approximation of it. Just a handful of

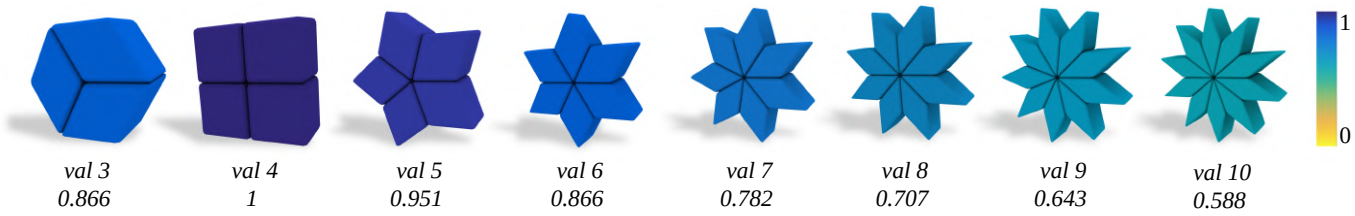


Fig. 7. Topology and geometry are tightly coupled: the number of hexahedra incident to a singular edge directly bounds the inner angles, thus affecting the geometric quality of the elements (numbers below each configuration refer to the Scaled Jacobian of the geometric map). Image from [Livesu et al. 2021].

methods provide strict guarantees on the maximum (Hausdorff) distance from the reference geometry, whereas in many cases an approximation of the input geometry without strict error control is produced. Depending on the complexity of the input shape, significant deviation from the target geometry may be present;

- **features:** special care must be taken for input features such as sharp creases. While the general requirement is the same as for geometric fidelity, imprecision in the geometric approximation of features is both aesthetically much more evident and may also have a significant impact in the solution of the PDE (e.g., when studying the aerodynamic flow around creased objects). Feature alignment requires that sequences of edges of the hex-mesh conform to feature curves, otherwise some deviation is inevitable, regardless of resolution (Fig. 8);

- **quality:** the assessment of the quality of a mesh is a major topic by itself [Knupp 2007] that is only touched upon in this survey (Sec. 3). It is important to note that the relation between mesh quality and, e.g., the quality of a numerical solution of a PDE may heavily depend on the concrete problem as well as on the solver at hand. While a common requirement is that all mesh elements are *valid* (everywhere positive Jacobian determinant of the geometric map), different numerical schemes may demand the fulfillment of additional requirements. Shape regularity criteria for the Finite Element Method (FEM) are mostly concerned with star-shapedeness and avoidance of large angles [Ciarlet 2002; Shewchuk 2002; Zlámal 1968]. As recently shown, these methods can be modified in order to even cope with badly shaped elements, locally selecting higher order basis that compensate for the lack of geometric quality [Schneider et al. 2018]. In Computational Fluid Dynamics (CFD) it can be beneficial to use meshes that are *orthogonal*, meaning that the interface between two shared elements and the line connecting their centroids form a right angle [Aqilah et al. 2018; Moraes et al. 2013]. The Virtual Element Method [Beirão da Veiga et al. 2014] assumes that all mesh faces are planar. Considering this jungle of metrics that are relevant for one numerical method or the other, general purpose algorithms are often not suited to address these specific criteria at the mesh generation stage, but mainly strive to create meshes with valid elements, possibly attempting to address further quality concerns in post processing (Sec. 6).

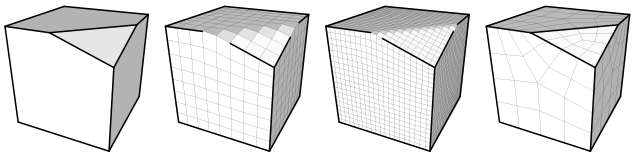


Fig. 8. Incorporating an input feature network (left, bold lines) into the output hex-mesh is not possible if the connectivity does not align to it (middle left), even refining the mesh (middle right). Key to feature preservation is the ability to align surface edges to the input network, carefully positioning mesh singularities (right). Image from [Livesu et al. 2020].

The methods surveyed in the following typically aim to create "good" meshes according to a subset of the criteria above. Fully and equally embracing both topological and geometric requirements at once can be a huge challenge, and many methods put a stronger focus on one aspect over the other. Some methods focus more on the topological aspects and may produce well structured meshes containing (near) degenerate or even invalid elements. Some others may guarantee valid elements or even lower bounds on certain geometric quality measures but produce meshes with a highly irregular topological structure. Both flaws can potentially be alleviated to some extent in post-processing, using dedicated algorithms for structure simplification (Sec. 5.5) or geometric enhancement (Sec. 6). Certainly, topology and geometry are coupled to some extent. For instance, a mesh with poor topological structure often inevitably contains poorly shaped elements as well (Fig. 7).

4.3 Advancing/Receding Front

First attempts to algorithmically generate hexahedral meshes were made by extending 2D advancing-front algorithms that generated full quadrilateral meshes. Starting from a quad-meshed boundary, algorithms like [Blacker 1996; Blacker and Meyers 1993] incrementally insert hexahedra starting from the boundary. The volume is progressively filled until final small voids are solved with simple patterns made of a few hexahedral cells. Such an approach is challenging on two main points. First, fronts can collide during their generation, and geometrical intersection must be performed. Owen and Sunil [2000] solve this issue by preserving a hybrid mesh during the whole process. Every created hex is inserted into this mesh, and front collisions are easily detected. The second point is much more problematic: there is no guarantee that the process will eventually generate a usable full hexahedral mesh. Starting from an even

number of quads on the boundary of a remaining void, a structural decomposition into a set of hexahedral elements is guaranteed to exist [Mitchell 1996], but the geometrical quality of hexes can be very low. And if one ends up with an odd number of quads surrounding a remaining void, one cannot fill it up with hexahedral elements at all, necessitating a backtracking of the front propagation (with no general guarantee to perform better the next time). The main reason for this inflexibility lies rooted in the fact that one cannot easily perform structural modifications on a 3D hexahedral mesh in a local manner (cf. Sec. 5).

Considering the problem as being over-constrained, the next generation of advancing-front algorithms do not start from a quadrilateral boundary mesh, but rather from the geometric surfaces [Staten et al. 2006, 2010a, 2005]. Complete layers of hexahedral cells are inserted in the domain until they collide. Final cavities are easier to fill, but this process can fail, too. In [Ruiz-Gironés et al. 2012], the authors adopt the advancing-front technique. Considering that the final cavities that remain may be difficult to mesh, they use an inside-outside mesh generation approach that requires as an extra input an inner seed, which is a hexahedral mesh of a possible final cavity. Two solutions of the Eikonal equation are then computed: one going inward from the boundary of the geometric domain; another one going outward starting from the surface mesh of the inner seed. Both solutions are then combined to define a smooth distance function, and an advancing-front algorithm is performed to expand the quadrilateral surface mesh of the inner seed towards the unmeshed external boundary using the distance function to locate points of each layer of cells. This process is used in practice to mesh the outside of objects like aircraft (for aerodynamics problems, for instance). But it remains limited to geometric domains that are homeomorphic with the sphere, and the domain must not have sharp features.

In general, advancing-front approaches are not reliable enough to generate a good quality hexahedral mesh for general domains. They strongly depend on the boundary mesh structure and the compatibility of this structure with the restrictive structure of hexahedral meshes. Often, this compatibility is not given, since the boundary mesh generation process is unaware of the structural and geometric constraints imposed by the to-be-created hexahedral mesh. As a consequence, they, e.g., fail to connect fronts when they collide (see Figure 9). Moreover, most of the proposed works deal with the extra constraint of starting from a pre-meshed boundary. This constraint is strongly related to the meshing process, which consists of meshing a complex assembly of parts where meshes must be conforming along part interfaces.

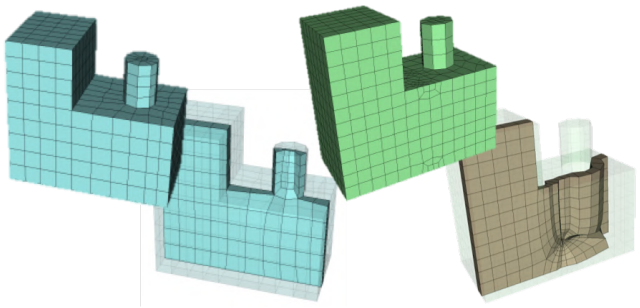


Fig. 9. Example of advancing-front progression to fill in a geometric 3D shape starting from different pre-meshed boundary. On the left, it succeeds in getting a valid hex-mesh, while it fails on the right. Image from [Ledoux and Weill 2008].

4.4 Dual Approaches

Taking a dual perspective in the context of mesh generation, i.e., focusing on the dual representation of a hexahedral mesh (Sec. 2.2), has proven to offer certain benefits.

Dual Advancing Front. For one, the interpretation of the advancing front approach (discussed in Sec. 4.3) in the dual domain can reveal interesting structures, simplify the formulation of constraints and rules, and provide additional intuition. This dual view is taken in the so-called Whisker Weaving [Tautges et al. 1996] method and its variants [Folwell and Mitchell 1999; Kawamura et al. 2008; Ledoux and Weill 2008]. These start from a prescribed surface quad-mesh that is to be matched by the hexahedral mesh to be constructed. Accordingly, the quad-mesh's dual loops form the prescribed boundaries of the hex-mesh's dual sheets. The algorithms' objective thus is to determine the dual sheets – in particular their mutual intersection combinatorics – inside these prescribed boundary curves.

The addition of a next hexahedron in the course of an advancing front approach can be interpreted in the dual as (combinatorially) fixing the intersection of three dual sheets [Tautges et al. 1996] or as locally (combinatorially) contracting one of the sheet boundary loops, conceptually fixing part of the dual sheet and leaving the loop as the boundary of that part of the sheet that is yet to be determined [Folwell and Mitchell 1999]. The dual view enables the formulation of local and semi-local rules and heuristics to more favorably steer the incremental mesh construction process [Folwell and Mitchell 1999; Ledoux and Weill 2008]. Nevertheless, issues such as poorly shaped elements, inverted elements, or high valence vertices in the result are not easy to avoid in general, even with this dual perspective.

A particular challenge for this approach is posed by the (very common) existence of self-intersecting dual loops in the prescribed boundary quad-mesh. While there is no general theoretical obstacle to the successful meshing of these, such loops need to be brought into pairwise or manifold correspondence and be filled by common sheets of non-trivial topology. It is unclear how the process can be steered to naturally establish this required structure in general; therefore, degenerate elements (so-called knives, Fig. 32) and inverted elements are common in the result in these cases. Various strategies (with more or less severe negative side effects on quality) have been proposed to modify the quad-mesh to get rid of such self-intersections in advance [Folwell and Mitchell 1999; Kawamura et al. 2008; Müller-Hannemann 2001; Müller-Hannemann 2002].

Dual Sheet-by-Sheet. Besides these alternative interpretations of advancing front methods, the dual perspective gives rise to a further class of methods, less local and incremental. A general challenge faced by algorithms that attempt to construct hex-meshes in an incremental fashion (like those discussed in Sec. 4.3) is to ensure

that "things work out in the end". Without careful look-ahead, one may easily end up in intermediate configurations that cannot be completed in either a valid or a qualitatively reasonable manner. Algorithms that, by contrast, approach the problem of mesh generation in a global manner, e.g., via global optimization formulations (cf. Sec. 4.8), on the other hand, can be computationally much more intensive.

The dual perspective permits an interesting incremental approach on a *semi-local* level. Instead of individual cells, entire dual sheets can be considered as the atomic entities for incremental mesh generation in the dual domain. For the case of quadrilateral mesh generation, which is in close analogy to the hexahedral mesh generation scenario, the advantages of this semi-local dual view for the purpose of incremental construction have been discussed in depth [Campen et al. 2012; Campen and Kobbelt 2014]. Similar properties hold in the hexahedral case, as is exploited by a number of algorithmic approaches. However, while in the quad case the dual is formed by chords, which are 1-manifolds (i.e., either a loop or a curve with two endpoints, possibly self-intersecting in points), in the hex case the dual consists of sheets, which are 2-manifolds of arbitrary genus and with an arbitrary number of holes, possibly self-intersecting in curves. Therefore, the problem is of significantly higher complexity and algorithms often restrict to sub-classes of problem instances for simplicity, such as objects of genus 0, sheets with a single boundary loop, or dual loops without self-intersections. An idea of inserting dual sheets in a divide-and-conquer manner was outlined by [Calvo and Idelsohn 2000]. A concrete algorithm for incremental hex-mesh construction based on sequential dual sheet generation is described by [Müller-Hannemann 2001]. The boundary geometry along an entire candidate sheet is assessed in the decision-making process. In contrast to related methods that can be interpreted as operating in a sheet-by-sheet manner [Folwell and Mitchell 1999; Ledoux and Weill 2008], this algorithm preserves an invariant through all intermediate stages that strictly avoids combinatorially invalid configurations. This obviates the need for intermediate repair operations and guarantees the absence of degenerate elements such as knives or wedges (Fig. 32). On the downside, the more restrictive sheet selection rules that are in place to ensure the invariant can bring the algorithm to an early halt. Rather expensive back-tracking strategies can be used as a remedy to some extent. By the introduction of additional rules for the selection of sheet operations [Kremer et al. 2014] in particular non-convex shapes can be handled in a more geometry-aware manner, commonly leading to less distorted (or less inverted) mesh elements.

Free Boundary. The above methods assume that a quadrilateral mesh of the domain boundary is given, effectively as a starting point for the incremental construction. The ability to prescribe a boundary mesh can be seen as an advantageous feature in some scenarios (e.g., when adjacent domains are to be meshed separately but compatibly). In others, it rather is a limitation: it restricts the meshing algorithm from the set of all hex-meshes suitable for the domain to a (small) subset. Recently there have been first attempts to construct hex-meshes on a sheet-by-sheet basis *without* predetermined boundary structure. Instead, they exploit interactive user guidance along the domain boundary [Takayama 2019] (Fig. 10), or loosely follow principal curvature directions [Livesu et al. 2020] to construct loops which then serve as sheet boundaries. It is worth remarking that the latter method essentially outputs a subdivided version of the primal mesh that is implied by the sheets; this has the effect that the sheets appear as primal facet sheets in the output mesh. Nevertheless, conceptually both methods are to be viewed as dual approaches.

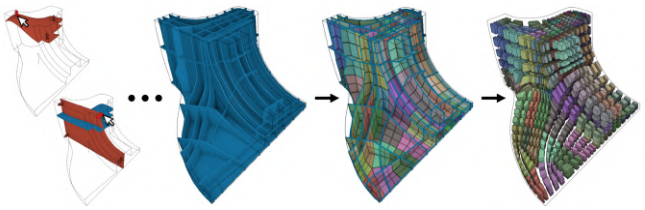


Fig. 10. Interactive sheet-based hex-mesh modeling. Image from [Takayama 2019].

Due to the larger search space compared to methods with prescribed boundary mesh, they conceptually have the potential to achieve results of better quality – but at the same time are computationally more expensive and require a user in the loop [Takayama 2019] or make simplifications sometimes leading to meshes that contain some non-hex elements [Livesu et al. 2020].

In this context, the interesting question is that of efficient geometric sheet representation – while in the above methods assuming a prescribed boundary-mesh, a non-geometric combinatorial representation was employed for simplicity. An implicit representation by means of a level set formulation has proven efficient [Takayama 2019]. It, however, does not support self-intersecting sheets, which would grant higher flexibility and enable better mesh quality in various cases. Another, discrete sheet representation space is described by [Roca and Sarrate 2008], embedding sheets in the facets of a particular tessellation of the domain; a concrete algorithm that operates in this space has not been addressed yet.

Dual Validity. Generally, when constructing hex-meshes out of dual sheets, it needs to be considered that not any arrangement of intersecting sheets implies a primal hex-mesh. A number of conditions need to be satisfied so as to avoid non-manifold configurations and self-adjacent elements, as detailed by [Mitchell 1996]. Violating sheet arrangements can be modified, often through the insertion of additional sheets, to ensure these conditions are met [Folwell and Mitchell 1999]. As these modifications not rarely have a negative impact on (geometrical and structural) mesh quality, a relevant challenge is to avoid the need for them right from the start.

4.5 Domain Decomposition

Early proposals for automatic domain decomposition relied on simple topological operations like submapping and sweeping [White et al. 1995], that were mainly trying to incorporate the knowledge of the users upon the two-dimensional domain to expand the decomposition to the third dimension with a sweeping step.

Sweeping. Given a volume represented by a closed surface, by identifying two patches where one serves as the *source* and the other

one as *target*, a hexahedral mesh can be generated through "sweeping" the quad-meshed *source* over the volume to the *target* [Shih and Sakurai 1996]. This simple idea is very suitable for CAD models since many shapes are formed by extrusion. The first batch methods using such an idea focus on shapes that can be easily meshed by identifying one source and one target, which are called one-to-one methods [Blacker 1996; Liu and Gadh 1997; Liu et al. 1999]. However, for slightly complex CAD models, more source or target patches have to be involved in decomposing the extrusion geometry into simpler one-to-one sub-volumes for easy processing.

A step ahead towards automatic decomposition is presented by Lu and colleagues [2001] that suggest recognizing in a CAD model the characteristics of portions that can be treated as submappable. The pipeline uses first a feature recognition, then a cutting plane identification, and, finally, a decomposition to mesh each portion with predetermined schemes. Along this direction, a set of many-to-one and many-to-many approaches are developed [Lai et al. 2000; Scott et al. 2006; White et al. 2004; Wu and Gao 2014; Wu et al. 2018]. These methods often rely on specific rules to detect line and planar features, such as various angle thresholds, so that the 3D model can be decomposed into sub-volumes having the same sweeping direction. If the decomposition is successful, various node insertion tricks for the sweeping can be employed to ensure the high quality of the generated hex-mesh [Knupp 1998; Ruiz-Gironés et al. 2011; Staten et al. 1999]. There are also approaches that allow multiple sweeping directions by computing a hierarchical sub-geometry structure [Miyoshi and Blacker 2000].

Kowalski and colleagues [2012] introduce the notion of fundamental sheets (fun-sheets), noticing that a hexahedral mesh is layered, in opposition to the lack of reference surfaces typical of tetrahedral meshes. Starting from a tet-mesh, converted in a hex mesh and identifying these fun-sheets, using topology and geometry of the shape, they obtain a better decomposition that catches the intrinsic characteristics of the shape. This approach is further enhanced in [Wang et al. 2017].

An interesting approach to the problem is the one presented by Lu and colleagues [2017]. They design and implement a sketch-based decomposition tool and evaluate its performance on a group of beginners and experienced users. They conclude that visual assistance and a geometric reasoning engine can help to obtain excellent results from a semi-automatic decomposition.

Medial Descriptors. Medial descriptors are a valid proxy in helping to realize a domain decomposition. Both mechanical objects and free-forms are possible to identify characteristics, mainly the skeleton catching the crucial elements of the shape's mutual relations. A three-dimensional shape's *skeleton* is, in fact, a topological representation of the shape capable of providing information regarding the various boundary entities' relative positions. The skeleton has been used in multiple methods to help in generating a hexahedral mesh inside the shape (see e.g., Fig. 11). When dealing with mechanical objects, usually containing boxes, it is vital to use the general skeleton (or medial object), including surfaces. Shapes more related to biology which can be approximated with a collection of generalized cones can be easily represented by their curve-skeleton (or medial axis). Both these proxies have been used to guide the hex-meshing.

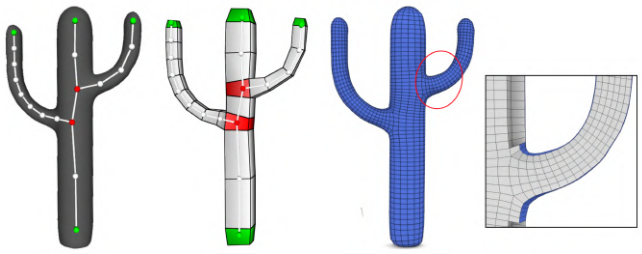


Fig. 11. Skeleton driven hex-meshing starts from an input surface mesh and line skeleton (left), around which a tubular structure composed of hexahedral boxes is initialized (middle left). Refining this structure and projecting it on the target surface yields a hexahedral mesh (middle right) where the distribution of the mesh elements aligns with the skeleton guiding curves (right closeup). Image from [Livesu et al. 2016].

Price and colleagues introduced the possibility to use the topological skeleton of the shape to produce a hex-mesh. They apply it first on convex shapes [Price et al. 1995], and then on solids with flat and concave edges [Price and Armstrong 1997]. The idea is to decompose the domain so that each sub-domain can be hex-meshed using a midpoint subdivision scheme [Li et al. 1995]. Each sub-domain is meshed using basic primitives that can be placed using the skeleton and used as elementary blocks to mesh the original domain. The topological information guides the choice of the correct primitive. There are limitations in the approach since high-valence boundary vertices do not have elementary schemes placing them.

Instead of using the skeleton, Sheffer and colleagues [1999] start from the embedded Voronoi graph of the domain, which is simpler to create. Using a set of configurations that include the Voronoi graph's local topology, it can decompose the domain in sweepable subdomains that can be combined and smoothed to yield the final decomposition of the whole domain. Through the computation of a harmonic field, a general 3D model can be decomposed into 2D curved slices where quad-mesh templates can be used to form a large structure decomposition of the 3D model [Gao et al. 2016].

Zhang and colleagues [2007] exploit the particular shape of the vascular structure to devise a method that uses the curve skeleton as a basis for the meshing. It is the first proposal in which there is decomposition in tubular subdomains that are quite simple to mesh via sweeping. The uniform diameter of the typical vases treated in the application does not pose the problem of resolution in the elements. Usai and colleagues [2015] use the curve-skeleton to derive a quadrilateral base complex given the triangular mesh of shape. The surface decomposition can be expanded to the domain's interior and lead to a method for hex-meshing [2016]. In this work, a scheme for keeping the mesh elements uniform while the diameter of the subdomains changes is introduced and applied. Another similar approach [Livesu et al. 2017] employs solid cylindrical parameterizations to map from the curve-skeleton to the cylindrical subdomains. This choice allows a simple but effective way to use the topological information to generate the hex-mesh.

All the methods described in the previous paragraph work fine only for models resembling collections of generalized cones.

Quadros [2014] also uses the skeleton as a starting point for meshing and, combining it with an advancing front approach, can create hex-dominant meshes. The surface and the skeleton jointly contribute to form what the author calls corridors that are the basis for meshing the domain with an advancing front method.

Cai and Tautges [2015] propose an approach that heavily relies on integer programming due to the classification of the edges for their parameterization. It is in line with the topological methods since it introduces a new set of templates that, once applied to the class of objects they use in their experiments: mechanical parts.

Another interesting approach [Liu et al. 2015] mixes skeletal representation of the shape and polycubes to guide the creation of the hex-mesh. The resulting meshes are non-conforming, including T-junctions. Another type of non-conforming decomposition, the so-called motorcycle complex [Brückler et al. 2022b], can be constructed guided by a seamless parametrization (cf. Sec. 2.5). This decomposition has hexahedral subdomains only, and can be refined into a conforming hexahedral mesh.

Once a suitable decomposition is computed, submapping or sweeping approaches can likely generate hex-meshes with satisfactory quality. For example, [Wu et al. 2017] can be employed to first generate a quadrilateral mesh for interfacing surfaces while ensuring conformity among adjacent sub-volumes, and then apply the straightforward sweeping to generate the final hex-mesh. However, up to now, the critical issue still lies in how to robustly decompose a 3D model into sweepable sub-volumes while ensuring the necessary conformity and feature preservation at the interfaces of different parts. Industry resolved this issue by putting the user in the loop, relying on manual block decomposition and automatic sweeping as a workhorse for hex mesh generation in commercial software [Altair 2022; ANSYS 2022]. Automatizing the process and freeing the user from tedious critical work is an open challenge for future methods in this family.

4.6 Grid based

A hex-mesh can be trivially created by voxelizing the interior of a closed surface and then projecting its boundary onto the target geometry [Schneiders 1996a]. Geometric fidelity can be controlled by tuning the resolution of the voxelization. Since the size of regular grids grows cubically, to reduce element count a set of adaptive spatial partitioning approaches that rely on hierarchical structures have been proposed. However, adaptive grids do not define a conforming hex-mesh because adjacent grid elements may have different size, generating spurious (*hanging*) nodes. Grid-based methods differ to each other for the refinement policy they use, for the technique used to suppress hanging nodes, or for the method used to project the mesh on the target geometry.

Methods in this class are among the firsts that were introduced in the field. From a mesh quality standpoint, they are typically considered inferior to other methods because: (i) the grid is fixed in space and the result depends on the orientation of the model; (ii) the connectivity they generate is intricate and rich of singular edges with high valence [Livesu et al. 2021]; (iii) the meshes they generate are highly unstructured and do not endow a coarse block decomposition (see Fig. 1 and Fig. 21 in [Livesu et al. 2020]). Nevertheless, when compared with alternative options grid-based methods really stand out in terms of robustness. To date, they are the only fully automatic methods capable of successfully hex-meshing any input shape, regardless of its geometric or topological complexity. For this reason, they are the only automatic methods currently implemented in professional software [CoreForm 2022a; CUBIT 2022; Distene SAS 2022]. Despite the most prominent methods were developed more than 10 years ago and the field remained quiet for some years, major improvements have been proposed in recent years, also opening avenues for further research.

Refinement. Grids should satisfy both local and global criteria. At a local level, cell size must be compatible with the local size of the input object, ensuring geometric fidelity. At a global level, it must be possible to select a subset of grid elements (e.g., the ones completely internal to the input shape) such that the topology of this arrangement matches the one of the original object. In case the grid and the input mesh are not homotopic, a bijective mapping between them is not possible. Local criteria are easier to enforce. The most typical split rules used in the literature are normal similarity [Ito et al. 2009], local thickness [Livesu et al. 2021; Maréchal 2009; Pitzalis et al. 2021], surface approximation [Gao et al. 2019] or a combination of these and other indicators [Bawin et al. 2021]. The fulfillment of global criteria is more complex and demands to preprocess the input shape [Mitchell and Vavasis 1992]. For this reason, the vast majority of methods do not guarantee that the output hex-mesh will have the same genus and number of connected components of the input model [Livesu et al. 2021; Maréchal 2009; Pitzalis et al. 2021], or ensure this property at the cost of severe over refinement (e.g., iteratively splitting all grid elements until topological equivalence is obtained [Gao et al. 2019]). Refined cells can be split in two alternative ways: *2-refinement* splits each edge in two, thus obtaining 8 sub-cells for each adjacent hexahedron; *3-refinement* splits each edge in three, thus obtaining 27 sub-cells. In both cases, the sequence of splits is encoded in a hierarchical tree structure, which corresponds to an octree for the 2-refinement, and to a 27-tree for the 3-refinement. Approaching this body of literature for the first time may be confusing, because all methods generally refer to these data structures as "octrees", even though this is not always correct. The use of 27-trees for 3-refinement is explicitly mentioned in [Schneiders et al. 1996] and a few other articles, and is only implicitly assumed in other articles that refer to these ones.

Hanging Nodes. The removal of hanging nodes is obtained by substituting elements of the grid with templated topological transitions that locally restore mesh conformity (Fig. 12). If adjacent grid elements differ by at most one level of refinement there exist 2^8 alternative configurations which, discarding symmetries, reduce to 20 unique cases [Weiler et al. 1996]. Existing methods can be broadly categorized into two families: *primal* methods aim to directly incorporate the hanging nodes in the output hex-mesh; *dual* methods aim to modify the input grid such that its dual mesh contains only hexahedral cells.

Primal methods often operate on 3-refined grids and 27-trees, because it is easier to suppress their hanging nodes [Schneiders et al. 1996]. However, handling all the possible 20 configurations is provably impossible, because many concave transitions are bounded by

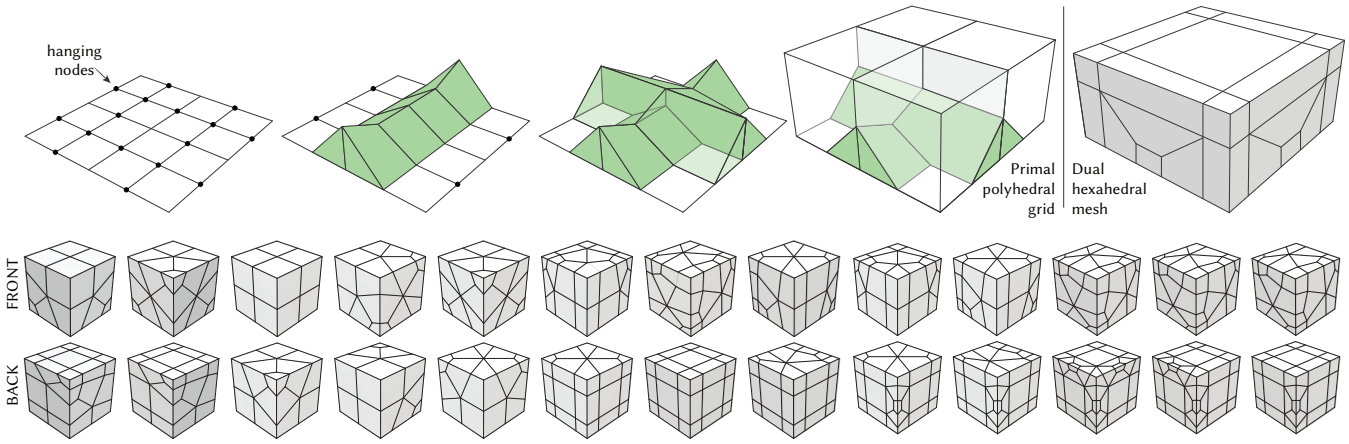


Fig. 12. Dual methods regularize the valence of hanging nodes (black dots) by connecting them pairwise along triangular bridges, so that the dual is a hex-mesh. Top: the flat transition firstly introduced in [Maréchal 2009]. Bottom: the set of atomic schemes introduced in [Livesu et al. 2021] to handle all possible transitions in strongly and weakly balanced grids. Images partly from [Livesu et al. 2021].

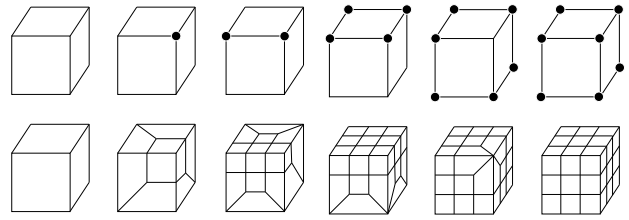


Fig. 13. A subset of the 3-refinement schemes introduced in [Schneiders 1996b]. The leftmost and rightmost elements correspond to the unrefined and fully refined cubes, respectively. The other templates are used to manage the transition between them.

an odd number of quadrilateral elements, a condition for which it is known that a hexahedralization of the interior does not exist [Mitchell 1996]. Transition schemes for 4 flat and convex transitions (see Fig. 13) appeared in multiple articles [Schneiders 1997, 1999, 2000; Tack et al. 1994] and were successfully used to compute hexahedral meshes, prescribing additional refinement to convert unsupported transitions into the supported ones. Over the years additional schemes were introduced to handle concave edges [Elsheikh and Elsheikh 2014; Ito et al. 2009; Zhang and Bajaj 2006], but a correct handling of concave corners remains elusive. Several works, like [Ebeida et al. 2011; Owen et al. 2017; Zhang et al. 2013], exploit the 2-refinement schemes introduced in [Schneiders et al. 1996] to remove hanging nodes. Unlike from the 3-refinement approaches, the grid needs to satisfy more strict constraint as those described below for dual methods. Note that, as for the 3-refinement case, the schemes in [Schneiders et al. 1996] do not allow to address all the possible configurations, often leading to an excessive over-refinement of the grid.

Dual methods operate on 2-refined grids and octrees, and are superior to primal methods because they can handle all possible transitions. All known schemes operate on *balanced* grids, that is, grids where the refinement mismatch between adjacent elements is at most one. However, not all methods agree on the definition of "adjacent". For the majority of methods two cells are adjacent if they share one face, edge or vertex (*strong balancing*). In [Livesu et al. 2021] the authors relaxed this formulation, enlarging the class of balanced grids and limiting restrictions to size mismatch only for cells sharing a face (*weak balancing*). Weakly balanced grids permit to greatly reduce refinement (up to 64% less elements in their experiments), but require a slightly more complex scheme set. Maréchal was the first to observe that if all grid vertices have valence 6 and all grid edges have valence 4, the dual of the grid is a pure hexahedral mesh [Maréchal 2009]. Based on this observation he proposed a set of cutting schemes that, regularizing the valence of grid elements, allow to obtain a pure hexahedral mesh via dualization (Fig. 12).

Since the valence of hanging nodes is fixed pairwise, dual methods also require that the grid is *pair*, that is, for each cluster of grid elements with same amount of refinement the number of hanging nodes must be even across all grid directions. Differently from balancing, the pairing condition is non local, hence difficult to enforce. Pairing is typically enforced directly in the octree, fully splitting parent nodes if their siblings have been split [Gao et al. 2019; Hu et al. 2013; Livesu et al. 2021; Maréchal 2009]. As shown in [Pitzalis et al. 2021] all these methods operate in a restricted space of solutions and tend to severely over refine the input grid, even if it is already pair. The authors showed that pairing can be enforced directly in the grid by solving a sequence of linear problems, obtaining coarser grids that approximately halve the number of elements. Despite superior to tree-based methods, also this method does not cover the whole space of solutions, and may occasionally refine an already pair input grid (see Sec. 7 in [Pitzalis et al. 2021]). Even though dual approaches exist since 2009, the transition schemes they use were only vaguely described in the literature, making these methods hardly reproducible. Maréchal [2009] pioneered this technique, but his paper describes in detail only one specific transition (Fig. 12, top). Gao and colleagues proposed three alternative schemes based

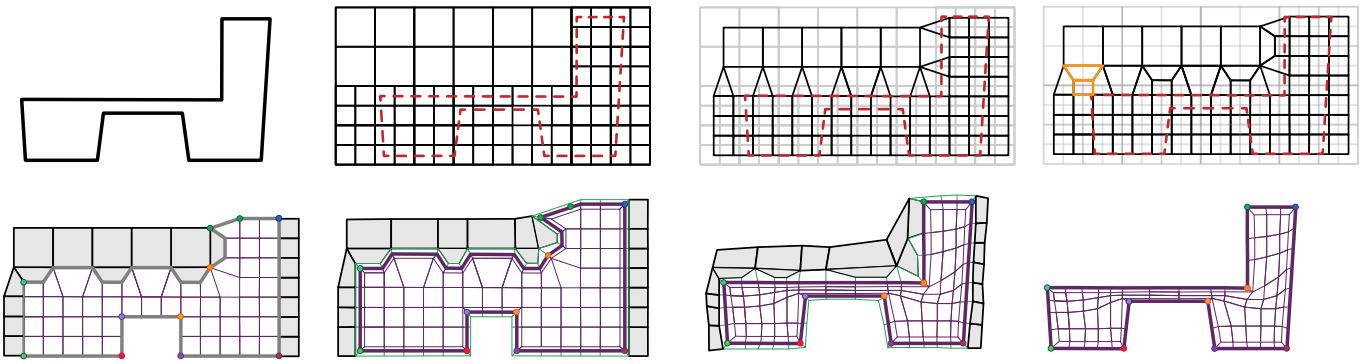


Fig. 14. 2D pipeline of the feature preservation octree-based hex-meshing. Top row: adaptive quadtree constructed from the input, dual of the quadtree, and quadrilateral (quad-) mesh including a scaffold mesh. Bottom row: topological matching of feature graphs, variational padding of both the target mesh and the scaffold, mesh deformation to fit the input, and the final pure quad-mesh. Image from [Gao et al. 2019].

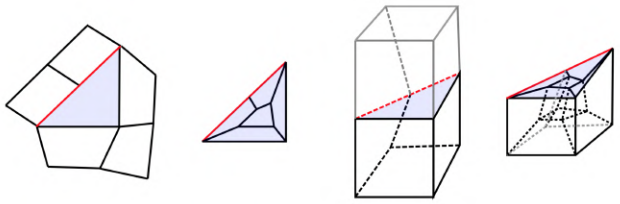


Fig. 15. If a quad maps two of its four edges onto a linear feature line it becomes locally degenerate (left). Splitting it into 5 sub-quads ensures enough degrees of freedom to produce all valid mesh elements. Similar configurations may also occur on 3D meshes, and can be resolved with a special padding scheme that splits a hexahedron into 6 sub elements (right). Image from [Gao et al. 2019].

on similar ideas [Gao et al. 2019], also releasing their code, but these schemes were recently shown to be not fully exhaustive and may fail to produce a conforming hex-mesh starting from a balanced and paired grid [Livesu et al. 2021]. In [Livesu et al. 2021] the authors propose a comprehensive study of dual schemes, clarifying ambiguities and implementative choices, and ultimately deriving an exhaustive optimal set of transitions for both strongly and weakly balanced grids (Fig. 12, bottom). CinoLib [Livesu 2019] hosts an open source implementation of all such schemes, as well as the code necessary to install them in a given adaptive grid.

Projection. Considering the axis-aligned nature of grid-based methods, to approximate the input object well the boundary vertices have to be projected onto the target geometry. To this end, maintaining the inversion-free property of a hex-mesh poses a great challenge. While [Lin et al. 2015; Maréchal 2009] rely on iterative vertex smoothing to slowly move the vertices onto the boundary so that a local smoothing can be backtracked if it causes flipped hexahedra, [Gao et al. 2019] presents a global deformation method that can robustly align the generated hex-mesh with the input surface (including sharp features) within a distance bound. Fig. 14 shows the 2D pipeline of the method presented in [Gao et al. 2019]. After grid refinement and removal of hanging nodes, the grid is partitioned into two sub-meshes: an inside "target" mesh that will be optimized to be the final output, and an outside "scaffold" mesh that ensures the bijectivity of the map throughout the optimization process. Geometric fidelity is achieved by first building a topological bijectivity mapping between the input mesh and the boundary of the target mesh, and then geometrically deforming the target mesh towards the input surface shape using a locally injective mapping technique [Rabinovich et al. 2017]. Note that a variational padding technique (see Sec. 5.4) is also introduced for both the target mesh and the scaffold, so as to increase the number of degrees of freedom for optimization. The approach can robustly produce an all-hexahedral mesh with several guarantees: 1) the output is manifold and its boundary surface has the same genus with the input, (2) all hexahedral elements have positive scaled Jacobian (3) the boundary of the hex-mesh is error-bounded, i.e., within ϵ distance from the input mesh, and (4) the boundary of the mesh has no self-intersections thanks to the scaffold mesh. All of this is obtained by trading robustness for efficiency, thus computational cost and memory resources can be prohibitive for commodity hardware. On the other hand, iterative methods such as [Lin et al. 2015; Maréchal 2009] are quite efficient, although may occasionally fail to preserve the shape well. Further research is needed to devise an algorithm that optimally combines robustness, efficiency and geometric fidelity.

Features. The preservation of sharp surface features is both geometrically and topologically challenging for grid-based approaches. First of all, since the mesh connectivity is derived by the underlying grid, surface vertices may not have enough incident edges to reproduce high valence feature points in the target mesh. Therefore only a subset of all possible feature networks can be faithfully reproduced. Moreover, hexahedra that have more than one facet exposed on the surface may easily be traversed by feature lines across more than one edge, becoming ill-shaped or even degenerate once projected onto the target geometry. To make sure that each element has at most one feature edge, specific padding schemes are used (Fig. 15 and Sec. 5.4). Finally, despite the fact that it works well in most cases, current algorithms for feature mapping are heuristic and do not offer guarantees. The most recent methods are based on ideas expressed in [Gao et al. 2019], and operate by iteratively processing each feature separately, projecting its endpoints to the

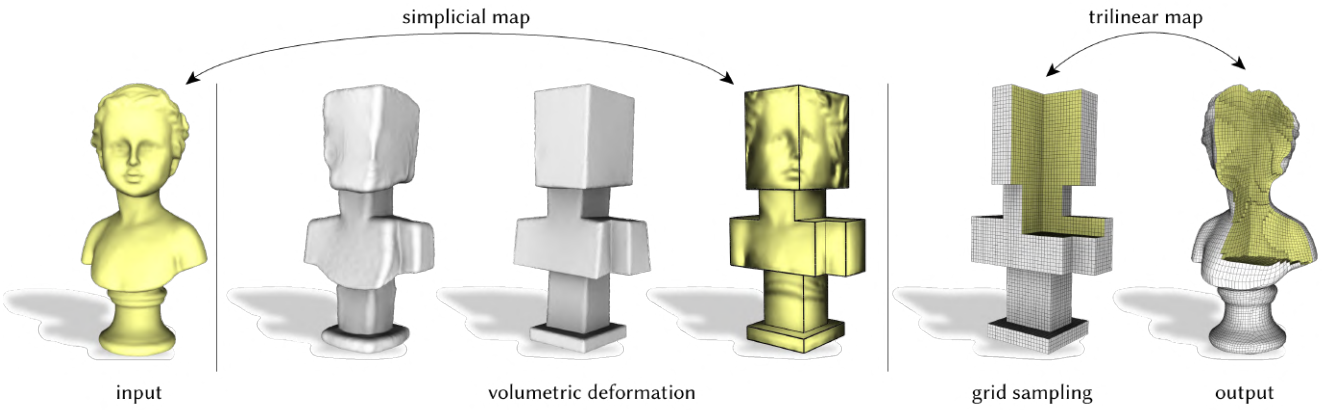


Fig. 16. The pipeline for polycube based hexmeshing generates a locally injective simplicial map through volumetric deformation, and then uses it as a medium to transfer a regular grid sampling of the polycube to the target shape.

closest vertices in the hex-mesh, and then finding the discrete path that connects them with a Dijkstra search that operates on a scalar field that encodes the euclidean distance from the input feature. Depending on the ordering of the features and the combinatorial structure of the hex-mesh, there can be conflicting configurations where a path that connects the two endpoints of a feature and does not conflict with any previously inserted feature does not exist. Furthermore, even if such a path exists, there may be cases in which the previously inserted features force a path to deviate from its geometric target significantly.

Assemblies and Multiple Materials. While all methods described so far assume as input a single model composed of a single material, grid-based techniques have been successfully extended to the multi material case [Su et al. 2004; Zhang et al. 2010], and can also handle complex non manifold CAD assemblies [Qian and Zhang 2012]. From a grid processing perspective, these methods rely on the processing techniques described in the previous paragraphs.

4.7 Polycube Maps

A successful line of algorithms works by volumetrically mapping a shape into an orthogonal polyhedron (or *polycube* [Tarini et al. 2004]) embedded in $\mathbb{R}^3$ whose corners align with the integer grid $\mathbb{Z}^3$. The integer grid inside the polycube then defines an (interior-regular) hexahedral mesh connectivity. Its nodes can be pulled back into the input object following the inverse map (Fig. 16), defining a hex mesh for the object. In this sense, this approach considers a special case of integer-grid maps (Sec. 2.5), further discussed in Sec. 4.8: the interior of the shape is restricted to be free of map singularities, thus free of irregular edges and vertices in the implied hexahedral mesh.

Polycube methods are therefore based on two fundamental building blocks: the definition of the polycube domain shape, and the generation of the volumetric map onto it. These two objectives can be pursued separately (i.e., defining a valid polycube domain structure first, and then computing the map) or together, letting the domain shape evolve while optimizing the map for low distortion and boundary alignment with the coordinate planes. The latter can be viewed as (incrementally) deforming the shape in a volumetric manner, aiming to find the polycube domain shape best fitting the input object.

Structure. The structure of the polycube can be defined by assigning to each surface element of the input (tetrahedral) mesh a label that represents one of the six global axes ($\pm X, \pm Y, \pm Z$). Clusters of adjacent elements with the same label identify the facets of the polycube. Various approaches have been pursued to assign labels, from purely local approaches, assigning to each surface element the axis closest to its normal [Gregson et al. 2011], over approaches taking context into account, e.g. using a modified centroidal Voronoi tessellation in the space of normals [Hu and Zhang 2016], to incremental approaches [Mandad et al. 2022]. The idea is to, afterwards, volumetrically deform the input mesh such that each surface element attains an orientation that corresponds to its assigned label.

However, not every labeling permits a corresponding polycube. Some correction procedures have been proposed, to be used as a postprocess or interleaved with the labeling [Gregson et al. 2011; Livesu et al. 2013]. A set of local conditions is known that allow checking whether the graph formed by a labeling corresponds to the graph of some orthogonal polyhedron [Eppstein and Mumford 2010]; they can be used to design label modification strategies [Livesu et al. 2013]. However, these conditions are neither necessary (they focus on a restricted set of polycubes) nor fully sufficient: suitability of the graph formed by the labeling does not imply suitability of the labeling itself, because the graph's *embedding* is ignored, as pointed out, e.g., by Mandad et al. [2022]. Sufficient conditions are of inevitably global nature, such as those considered by Sokolov [2016], who describes a complex post-process procedure to modify a labeling into a state structurally suitable for a polycube. Interactive tools for user assisted polycube construction or modification also exist [Li et al. 2021; Yu et al. 2022; Yu and Wei 2020].

Mapping. The volumetric map can be obtained using dedicated deformation energies that iteratively deform the object such that surface normals rotate until they *snap* to the global coordinate

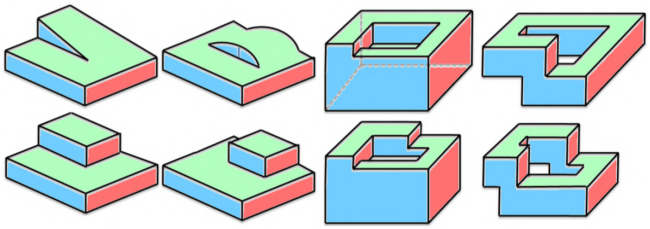


Fig. 17. Top: a set of pathological shapes with exemplary surface labelings that will push polycube deformation energies towards the generation of locally or globally inconsistent states, not defining a proper polycube. Bottom: after strategically modifying the labeling, a deformation into polycube shape is successful. Image from [Sokolov 2016].

axes. These methods may take a pre-computed polycube labeling as input [Gregson et al. 2011; Livesu et al. 2013], freely deform the shape until the polycube structure reveals itself [Fang et al. 2016; Huang et al. 2014; Mandad et al. 2022], or interleave the two operations, updating the reference labeling after each iteration [Fu et al. 2016]. These pipelines often include heuristic post-processing steps that aim to remove structural artifacts (e.g., removing label regions with less than 4 neighboring regions). In [2015] Sokolov and Ray point out that there are local as well as global conditions regarding structural validity, hence complete sanitization can be difficult. In particular, since axis-aligned features are quite naturally preserved during deformation, the presence of long, slightly diagonal creases may easily result in globally inconsistent configurations which are hard to recover from (Fig. 17). Tiny features such as protuberances, tunnels, and handles are also critical, and may even require input mesh refinement to enable any valid labeling.

Polycube deformation operates on a supporting tetrahedralization of the object. Early deformation energies defined in [Gregson et al. 2011; Huang et al. 2014] did not sufficiently penalize distorted, degenerate and flipped elements, producing maps that are not locally injective, especially in the vicinity of concavities. Fu and colleagues [Fu et al. 2016] introduced a deformation energy that incorporates the AMIPS term [Fu et al. 2015], which grows to infinity in the presence of degenerate or inverted elements. Various similarly flip-preventing energies have been introduced in recent years [Fu and Liu 2016; Rabinovich et al. 2017], and could be adopted in this setting. It is important to note, however, that the strict prevention of flips may reduce the deformation space to an extent that no map that respects the boundary-alignment constraints can be found, unless further mesh refinement capabilities are introduced.

Alternatively to volumetric deformation, one can in principle use a surface-based method to define a polycube-surface map (e.g., with [Yang et al. 2019]) and then solve for a compatible volumetric mapping between the two shapes. Again, however, despite the high level of practical robustness showcased by recent approaches [Du et al. 2020; Garanzha et al. 2021], the fully reliable automatic generation of constrained volumetric maps without flips remains an open problem [Fu et al. 2021]. Motivated by this difficulty, an interactive polycube construction pipeline that puts the user in the loop has been recently proposed [Li et al. 2021]. Users are allowed extensive control over each stage, such as editing the polycube structure, positioning vertices, and exploring the trade-off among competing quality metrics, while also providing automatic alternatives. The flip-averse mapping energy proposed in [Garanzha et al. 2021] is internally used to discourage the generation of flipped elements.
The use of alternative mesh representations has also proved useful to robustly construct volumetric mappings. In [Paillé et al. 2015] the authors represent a tetrahedral mesh as a collection of dihedral angles, and propose a robust spectral reconstruction method to generate an explicit mesh up to a global similarity transformation. The use of reduced coordinates to represent and manipulate meshes (e.g. via curvature or edge lengths) is a broad topic and has been widely studied, especially for the surface case [Campen et al. 2021; Crane et al. 2011]. Specifically, the aforementioned paper shows that any input polycube segmentation can be translated into a set of prescribed dihedral angles that encode the change of normal orientation along the surface. Using the proposed reconstruction method allows to convert such angles into an explicit mesh, obtaining a locally injective polycube map.

Quantization. Beyond piecewise aligning the object's surface with the coordinate axes, these pieces furthermore need to be aligned specifically with integer coordinates. Only then does each cubical cell of $\mathbb{Z}^3$ lie either entirely within or without the polycube domain, thereby implying a proper hexahedral mesh. The selection of the integer coordinates is sometimes referred to as *quantization*. A common strategy is to, in a first phase, generate a map ignoring the integer requirement, and then determining integer choices based on this *relaxed* solution, followed by a further deformation to match these choices. A classical approach for the determination of reasonable integers is *rounding*: for each planar surface region of the relaxed solution, select the integer closest to its constant coordinate. As this simple approach is fragile (especially for coarse target resolutions, incompatible integers obtained by rounding may force the map into degeneration), dedicated quantization strategies have been devised [Chen et al. 2019; Cherchi et al. 2016; Protais et al. 2022]. A recent quantization method [Brückler et al. 2022a], based on the so-called motorcycle complex [Brückler et al. 2022b], prevents degeneration altogether and is formulated for general integer-grid maps, of which polycube maps are a special case. It can therefore also be used in the context of frame-field based map generation methods, discussed in Sec. 4.8.

After quantization, the integer grid cubes contained in the polycube domain can be mapped into the input object via the inverse map, obtaining a hexahedral mesh. Depending on the curvature of the map, the individual hexahedra may undergo severe deformation. For instance in the common setting of trilinear hexahedra (cf. Sec. 3), this can lead to elements with flipped orientation, even if the polycube map is injective. Untangling techniques may be applied in such cases (Sec. 6), albeit without guarantees of correctness. The resolution of the integer grid inside the polycube (the "sampling frequency"), which can be controlled by appropriately scaling the map before quantization, of course has a significant effect on the likelihood of such issues.

Adaptive Resolution. While a regular grid of constant resolution (i.e. $\mathbb{Z}^3$) is typically used, adaptive sampling schemes can be used to control hexahedral element size and anisotropy. Adaptively sampled

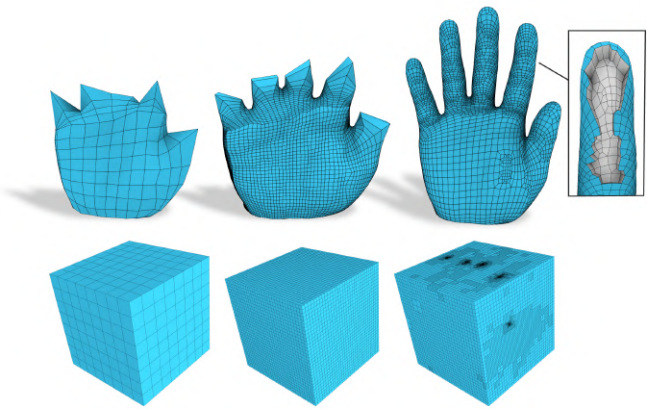


Fig. 18. Adaptively sampling a coarse polycube allows to restore major features that are not explicitly encoded in parametric space (right), and that could not be obtained with a regular sampling (left), not even with a dense one (middle). Image from [Pitzalis et al. 2021].

polycubes can be used to improve geometric fidelity while keeping the mesh resolution low, and to better capture regions of high distortion or curvature in the polycube map, restoring major surface features that are missed when using a regular sampling, unless extremely dense (Fig. 18). In recent literature there are attempts to address size control, obtained either by thickening a region of interest in polycube space prior to sampling [Xu et al. 2017] or adapting octree-based meshing to polycube space [Hu and Zhang 2016]. Pitzalis and colleagues [2021] showed how adaptive sampling can be unlinked from rigid octree hierarchies and extended to generalized grids of any shape or topology. These preliminary results suggest that a tighter integration of adaptive sampling in polycube space could benefit the whole pipeline. Current methods heavily rely on the ability of the polycube generation module to capture all the features of the object (at all scales) so as to secure a high quality mesh structure. Finding a better balance between features that are explicitly encoded in the polycube and features that will be reproduced with adaptive sampling, algorithms may be able to better distribute the complexity throughout the whole pipeline, possibly increasing their robustness. On the negative side, any non-regular sampling introduces additional irregularities in the mesh, so as to enable the transition between regions of different resolution. This reduces the level of structural regularity (Sec. 2.4) and can make the resulting hexahedral mesh unusable for applications that exploit a regular interior, or a coarse block decomposition endowed in the mesh connectivity, such as IGA methods [Hughes et al. 2005].

Features. Desirable properties such as curvature and feature alignment depend on how the polycube map orients these entities in $\mathbb{Z}^3$. In particular, since sharp creases are preserved only if they map to integer isolines in polycube space, there are intrinsic topological limitations to the class of feature networks that can be correctly reproduced (e.g., a convex vertex with more than three incoming feature lines cannot be correctly meshed). Since geometric features are often characterized by surface normal discontinuities, labeling methods such as [Hu and Zhang 2016; Livesu et al. 2013] intrinsically promote their positioning along polycube edges. Nevertheless, these methods do not explicitly handle surface features, and may often fail to preserve them [Guo et al. 2020]. To our knowledge, the only method that explicitly promotes feature alignment is CE-PolyCubeMaps [Guo et al. 2020]. Given an input network, the authors attempt to transform each feature into a piece-wise linear curve that aligns with the global axes. Features that do not align (or conflict with other features) are discarded; the others are included in the polycube structure generation, with a feature-aware variant of PolyCut [Livesu et al. 2013]. While practically superior to previous approaches, also this method does not provide strict guarantees. Furthermore, features are only mapped to polycube edges, and the possibility to align to integer isolines that are internal to polycube faces is not exploited.

Maturity. Polycube methods have received increasing attention from the meshing community and have now reached a decent maturity level. The most recent algorithms allow to blindly process datasets composed of more than a hundred shapes, producing hex-meshes of good quality [Fu et al. 2016]. In terms of mesh structure, these methods typically produce valence semi-regular meshes, and may occasionally produce semi-regular meshes if singularities (i.e., polycube corners) align [Cherchi et al. 2016]. The singular structure of a polycube-based hex-mesh is fully exposed on the surface, and consists of all polycube edges and corners. This inability to position singularities in the interior inherently limits the map, and may occasionally be the source of unnecessary distortion. A recent work of Guo and colleagues [2020] proposes to enhance the singular structure with diagonal cut surfaces that penetrate the interior of the polycube, permitting further distortion reduction. Intuitively, these cuts can be thought of as analogous to cone singularities in surface mesh parameterization [Soliman et al. 2018], although they are more constrained because the two copies of each cut surface must still obey to the constrained polycube structure. Another typical improvement consists in pushing the external singular structure one layer inside the volume, adding a global padding layer that avoids over-constrained hexahedra with more than one facet exposed on the surface (see Sec. 5.4). More sophisticated padding schemes that directly exploit the polycube map to optimally balance distortion with mesh growth are also available [Cherchi et al. 2019a].

Abstract Polycubes. Various techniques use segmentation (e.g. based on a shape skeleton) to partition a shape, generating an atlas of maps to a set of face-adjacent cuboidal domains [Li et al. 2010, 2013; Liu et al. 2015; Livesu et al. 2016; Usai et al. 2015] (closely related to domain decomposition approaches Sec. 4.5), or a map to a single but self-adjacent polycuboidal domain [Fang et al. 2016; Mandad et al. 2022], related by rigid transition functions. The latter can be viewed as an integer-grid map (Sec. 2.5) that may have transitions across parameter chart cuts, but these are restricted such that no interior singularities are implied. This provides additional degrees of structural freedom on shapes of higher genus. Sometimes these structures are referred to as *abstract* or *generalized* polycubes, emphasizing that these structures may not have a (continuous global) embedding in $\mathbb{R}^3$, due to the transitions. In contrast to standard polycubes, these – at least conceptually – enable the generation of any

hexahedral mesh with regular interior. For the generation of such non-embeddable polycubes, extrinsic deformation techniques, as described above, are typically ill-suited. Instead, intrinsic frame-field based integer-grid map generation (Sec. 4.8) may be used, restricted to the interior-regular setting [Fang et al. 2016; Mandad et al. 2022].

4.8 Frame Fields

Frame fields offer a promising research direction for general hexahedral mesh generation. A prototypical algorithm (see Fig. 19) consists of three major steps: (i) synthesis of a boundary-aligned frame field (Secs. 4.8.1 and 4.8.2), (ii) generation of an integer-grid map that resembles the frame field (Sec. 4.8.5), and (iii) extraction of the integer level-sets which form an explicit hexahedral mesh [Lyon et al. 2016].

The ultimate goal of such an algorithm is to find a valid integer-grid map (cf. Sec. 2.5) that minimizes a distortion objective while satisfying alignment constraints induced by boundaries or other features of the input geometry. Conceptually, it would be preferable to synthesize or optimize an optimal integer-grip map directly, without any intermediate frame field. Unfortunately, the underlying non-convex mixed-integer problem is too hard for available optimization techniques such that direct optimization without a good starting point inevitably results in a poor local minimum. Note that the first derivative of an integer-grid map $f : \mathbb{R}^3 \mapsto \mathbb{R}^3$ is a frame field of Jacobian matrices $J : \mathbb{R}^3 \mapsto \mathbb{R}^{3\times3}$. The idea of frame field based methods is to search for approximations of J, which are sufficiently accurate for identifying appropriate singularities (one discrete aspect of IGMs), while being significantly easier to optimize by ignoring various other difficulties of IGMs, e.g. integer quantization (another discrete aspect of IGMs), local injectivity, integrability, and element sizing. Consequently, a frame field can be understood as a *relaxation* of an integer-grid map.

One important goal when designing frame field schemes consists in finding a good tradeoff between faithfulness of the relaxation and ease of optimization. In fact, most frame field schemes further decompose the optimization task into different stages, e.g. first initializing the field with a rough but convex relaxation, and only subsequently continuing the optimization with a more accurate but non-convex formulation. State-of-the-art frame field methods differ in (i) the required input data (e.g. domain as triangle or tetrahedral mesh, or manual specification of singularities), (ii) the space of frames (e.g. octahedral, odeco, or general), (iii) the parametrization of frames (e.g. Euler angles, quaternions, 3×3 matrices, spherical harmonics coefficients, possibly in differential form), (iv) handling frame symmetries explicitly by matchings, or implicitly by lifting frames to a space with built-in symmetry, (v) the objective function (e.g. Dirichlet energy, or Ginzburg-Landau type energy), (vi) the optimization scheme (e.g. Gauss-Seidel relaxation, manifold optimization, or MBO with alternating diffusion/projection). All these variants are equipped with different advantages and drawbacks, which we will survey in more detail in the following while introducing the required background on-the-fly.

4.8.1 Frame Field Representations. A *frame field* can be seen as a generalization of a vector field to a quantity that locally describes the shape of a (linearly deformed) cube. Locally, a frame consists of three linearly independent vectors, which represent a parallelepiped, i.e., the orientation and shape of a linearly deformed cube. It is important to understand that globally a frame field is significantly more complex than three superimposed vector fields since it can contain singularities where topologically the vector fields are non-trivially interconnected on a branched covering [Nieser et al. 2011]. As a consequence, the connection induced by a frame field might exhibit nonzero monodromy, i.e., a vector does not return to itself when transported along a cycle around a singularity.

Space of Frames. There are various different representations to locally encode a frame. A straightforward choice that is capable of fully describing the shape of a linearly deformed hexahedron are three *explicit* vectors $u, v, w \in \mathbb{R}^3$ bundled into a matrix $F = (u, v, w) \in \mathbb{R}^{3\times3}$, called a *general frame*. The local shape of the hexahedron then simply corresponds to the parallelepiped formed by u, v and w. Usually, the space of frames is restricted to non-degenerate and orientation-preserving configurations, imposing the non-convex constraint $\det F > 0$. In practice, often subspaces of general frames are chosen, where additionally F is orthonormal (*octahedral frame*) or orthogonal (*odeco frame*), meaning that only rotations or rotations and scaling along the principal axes are possible.

Parametrization of Frames. For each space of frames – octahedral, odeco, or general – there are different parametrizations available, besides the matrix F. So far, octahedral frames have been investigated most extensively. They correspond to rotations, which can be parametrized by unit quaternions [Gao et al. 2017b; Liu et al. 2018], Euler angles, or an axis-angle representation. Euler angles have been used either w.r.t. a global coordinate system [Huang et al. 2011], or alternatively a local coordinate system [Palmer et al. 2020; Ray et al. 2016] to avoid gimbal locks. Adding three positive scaling factors to any of the octahedral frame parametrizations turns them into odeco frame representations.

Handling Cube Symmetries. The explicit representation of frames via linear transformations has one major disadvantage; it is not unique. For example, the matrix (u, v, w) transforms the unit-cube into an identical parallelepiped as the matrix $(v, w, -u)$. Since there are 6 potential permutations of the three vectors and 2^3 potential choices of sign, in total, there are 48 different matrices, which encode a single frame. Formally, equivalence is established by the binary octahedral group BO with 48 symmetry transformations. Since the octahedron is dual to the cube, their symmetry transformations are identical. By fixing the orientation (the sign of the determinant of F), it is possible to reduce the elements in one equivalence class to 24 elements with octahedral symmetry. This explains the term *octahedral field* [Solomon et al. 2017], which is often used for frame fields restricted to rotations, while *3D cross field* is yet another common name. The non-uniqueness of the (u, v, w)-representation significantly complicates the optimization of frame fields by inducing discrete variables κ, called *matchings*, between neighboring frames, with values from the octahedral group O. With *explicit* matchings a frame field algorithm needs to simultaneously optimize *discrete* matchings in addition to *continuous* frame degrees of freedom, e.g. [Gao et al. 2017b].

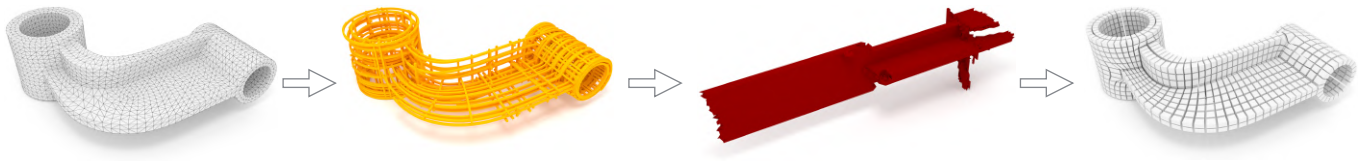


Fig. 19. Frame field based hexahedral mesh generation: Given an input tetrahedral mesh (left), first a boundary aligned and smooth frame field is generated (middle, yellow). The frame field serves as a proxy for the local orientation of hexahedra and thus enables the efficient generation of an integer-grid map (middle, red), which induces a hexahedral mesh (right). Image from [Liu et al. 2018].

An alternative to matchings are representations with a built-in symmetry, which offer a unique representation of equivalent frames. The work of Huang and colleagues [2011] expresses octahedral frames as rotations of the polynomial $x^4 + y^4 + z^4$, which are by construction invariant under transformations by elements of O. By restricting the polynomial to a sphere, it can be expressed in the spherical harmonics basis, lifting a single octahedral frame to a 9-dimensional representation vector. Since rotations only possess three degrees of freedom, it is clear that the spherical harmonics representation is a relaxation, i.e. not all 9-dimensional vectors correspond to a rotation of the polynomial $x^4 + y^4 + z^4$. An identical representation can be derived from the perspective of 4th-order symmetric tensors [Chemin et al. 2018; Golovaty et al. 2021], which have been further generalized to a 15-dimensional representation of odeco frames offering independent scaling of axes [Palmer et al. 2020], or even general, non-orthogonal frames [Desobry et al. 2021]. All these representations lift the frame representation to a (non-linear) sub-manifold embedded in a higher-dimensional coefficient space, which imposes challenges from the optimization perspective.

Differential Frame Representation. Instead of representing a frame field by pointwise specification of frames, one valuable alternative consists in encoding its derivative, i.e. the change of frames. After specifying one frame in the domain, the entire frame-field can then be re-constructed by integration. Such integration is path-independent given that the specified derivative is integrable, i.e. all fundamental monodromies are elements of the octahedral group when expressed in the coordinate system of the frame itself. Corman and Crane [2019] employ such a differential representation in a frame-field optimization setting with prescribed singularities. They extend the theory of moving frames to frame-fields with cube symmetry. Conceptually, the setting is analogous to the 2D setting addressed by Crane et al. [2010]. In both cases, all singularities and thus all fundamental monodromies need to be specified as input. However, while in the 2D setting a simple linear solve is sufficient to solve the optimization problem, the non-commutativity of 3D rotations requires a (continuous) non-linear least-squares optimization. Interestingly, despite the non-convex objective function, experiments suggest that the resulting field is independent from the chosen initial configuration. Leveraging a differential frame-field representation for optimizing fields with unconstrained singularities has not been done so far but is an interesting direction for future work. It would require replacing the fixed monodromies by the feasible set of discrete choices from the octahedral group.

4.8.2 Frame Field Optimization. The optimization problem usually consists in finding the "best" frame field, which aligns to the boundary of the domain. Best in this context is often interpreted as as-smooth-as-possible and is specified by an objective function. This objective on the frame field level essentially serves as surrogate for the actual objective of low distortion on the integer-grid map level, as well as, indirectly, for the objective of integrability of the frame field. Note that a good choice of objective function, constraints, discretization, and optimization scheme strongly depends on the frame representation at hand. Different combinations can lead to very different trade-offs between the faithfulness of the relaxation and the ease of optimization, as will be discussed in the following.

Objective Function. Smoothness of the frame field is the most widely employed objective function to approximately minimize the distortion of the integer-grid map. Consequently, most algorithms are based on a discretization of the Dirichlet energy $E_D = \int_\Omega ||\nabla\phi||^2 dx$, where ϕ is a frame representation. The space of frames and the chosen parametrization matter in this context. For instance, similarly to the 2D setting (cf. [Vaxman et al. 2016]) one observes different behavior when solely optimizing rotations (angle-based representations in 2D) in comparison to a mix of rotations and magnitudes (Cartesian representations in 2D). Typically, rotation-based objective functions are observed to lead to superior singularities in the sense of enabling lower distortion of the resulting integer-grid map. However, on the downside, rotation-based objectives are usually more difficult to optimize (due to a higher level of non-convexity), and they suffer from an energy blow-up at singularities, making them tessellation dependent (cf. [Knöppel et al. 2013]). A beneficial middle ground is offered by Ginzburg-Landau type energies. The idea is to work with a (convex) Cartesian representation $\phi \in \mathbb{R}^n$, which relaxes the (non-convex) manifold of intended frames $\mathcal{F}$, and to add a penalty term $E_P = \int_\Omega dist(\phi, \mathcal{F})dx$, which limits unintended behaviour in the relaxed space. The total objective $E_{GL} = E_D + \frac{1}{2\epsilon^2} E_P$ behaves like an angle-based scheme for ϵ approaching zero, while at the same time offering a convex relaxation when ϵ approaches ∞. Hence, ϵ continuously trades faithfulness of the relaxation versus ease of the optimization. Another advantage of Ginzburg-Landau type energies is that they can be optimized by the efficient and easy-to-implement MBO scheme, which will be discussed below.

Another important objective is the control on sizing of the hexahedral elements. The optimization of a smooth frame field typically results in singularities that are appropriate for uniform sizing. Non-uniform and anisotropic sizing can be added to frame-field-based

methods by either pre-deforming the domain[Xu et al. 2017] or by synthesizing a Riemannian metric field, and then optimizing the frame field w.r.t. this metric field [Fang et al. 2021]. Again, the decoupling of the metric field synthesis from the actual frame field optimization is suboptimal from an accuracy perspective, however, highly beneficial to ease the optimization in absence of a good starting point.

Constraints. Some earlier methods constrain the complete boundary field to a pre-computed solution [Kowalski et al. 2016; Li et al. 2012]. However, in general, it is preferable to only require boundary alignment and let the rest of the field emerge freely, including the frame directions tangential to boundaries [Huang et al. 2011; Palmer et al. 2020; Ray et al. 2016].

Discretization. Typically, the domain is discretized into a tetrahedral mesh, where frames are located at vertices [Gao et al. 2017b; Palmer et al. 2020; Ray et al. 2016], faces [Huang et al. 2011], or cells [Liu et al. 2018]. Alternatively, boundary element discretizations have been explored [Solomon et al. 2017], where a triangulation of the boundary, as opposed to a tessellation of the volume, is sufficient.

Optimization Scheme. Algorithms that are based on a lifted frame representation (9- or 15-dimensional) need to ensure that they do not leave the sub-manifold of desired frames. This is done either with a projection operator, or with some kind of manifold optimization. More specifically, Huang et al. [2011] compute an initial field by optimizing a convex relaxation of the actual problem, i.e., minimization of the Dirichlet energy in $\mathbb{R}^9$, followed by a local projection onto closest frames. The boundary alignment of the field is approximated by a single linear constraint per boundary face. The initial field is further improved by a nonlinear optimization restricted to the frame-manifold via Euler angles. Ray et al. [2016] follow a very similar strategy but discretize the field on vertices, tighten the boundary constraints and improve the performance of the projection. Palmer et al. [2020] observed that a modified Merriman–Bence–Osher (MBO) algorithm is beneficial because it is often able to avoid local minima that induce global inconsistencies in the singularity graph. The MBO algorithm alternatingly diffuses the 9- or 15-dimensional coefficient space and locally projects the coefficients onto frames. In 2D it is known that the MBO algorithm optimizes the Ginzburg-Landau energy [Beaufort et al. 2017; Viertel and Osting 2019], where the diffusion parameter is directly related to ϵ of E_{GL}. The 3D version behaves similarly, however, the mathematical theory has not been fully developed yet. Instead of the constant diffusion kernel of the standard MBO algorithm, the modified MBO algorithm starts with a large diffusion kernel and then iteratively shrinks it in subsequent steps. The rationale behind this strategy is that large diffusion steps sufficiently leave the (non-convex) manifold of frames and thus avoid local minima, while small diffusion steps are required for the accuracy of the solution. The diffusion parameter is directly related to ϵ of the Ginzburg-Landau energy discussed above. Hence, the modified MBO algorithm can be understood as slowly traversing from an easy-to-solve but less accurate relaxation to one that requires a good initialization but is more accurate. This explains the empirical observation that among all available option the modified MBO scheme of [Palmer et al. 2020] behaves best. The projection of frames in such an MBO framework can be done approximately with gradient descent [Ray et al. 2016], or exactly with a semidefinite relaxation [Palmer et al. 2020]. More rapid convergence than the MBO algorithm is offered by Riemannian trust-region manifold optimization [Palmer et al. 2020], which on the downside has a higher risk of getting trapped in local minima. Similarly to other highly non-convex schemes, it requires a careful initialization, e.g. by a convex relaxation.

4.8.3 Generality. Besides providing directional guidance for hexahedral mesh elements, a key property of a frame field is its network of singularities – which one commonly aims to adopt for a hexahedral mesh generated based on the frame field. In this context, frame fields are general enough that the singularity network of any hexahedral mesh can be expressed. This means that, in contrast to many other approaches, e.g. polycube mapping, sweeping, or grid-based approaches, the output is not a priori restricted to a subclass of hexahedral meshes. Therefore superior mesh quality can be achieved, specifically if complex feature alignment is required. In particular, frame field-based methods are able to express alignment not only to the boundary of a domain but also to arbitrary internal structures, which is, for example, important in multi-material applications or in the simulation of fluid-structure interaction.

4.8.4 Non-Meshability. The main drawback, on the other hand, is the fact that frame fields are actually over-general for the purpose of mesh generation: Frame fields may exhibit additional types of singularities that cannot occur in hexahedral meshes, cf. [Liu et al. 2018; Viertel et al. 2016]. Such singularity configurations are said to be "non-meshable". A key example are 3-5 singularities [Reberol et al. 2019], which frequently appear in smooth frame fields but are not meshable. Existing approaches are able to automatically repair some locally non-meshable configurations [Jiang et al. 2014; Li et al. 2012] or involve the user to manually repair the singularity graph [Liu et al. 2018] and then generate a frame field with a prescribed singularity network [Corman and Crane 2019; Liu et al. 2018]. Another option is to optimize a general frame-field in such a way that all singularities are pushed towards the boundary in order to generate a generalized polycube as done in [Fang et al. 2016]. Additional research is required to enable complete repair or to restrict the frame field generation and optimization to the space of meshable configurations in the first place. Little can be learned in this regard from the analogous 2D problem, as the gap, in terms of singularity structure, between 2D frame fields and quad-meshes is significantly smaller.

4.8.5 Field-Guided Integer-Grid Map. Given a (meshable) frame field, one then aims to conceptually *integrate* it to obtain a parametrization, a map (in particular an integer-grid map) onto part of $\mathbb{R}^3$. As the frame field typically is not integrable, a map whose isocurves are precisely aligned with the frame field's directions does not exist. Approximate alignment, e.g. least-squares alignment, is thus aimed for, as in the Poisson approach described by [Nieser et al. 2011], generalizing the cross-field guided mapping used in the 2D case for quadrilateral meshing [Bommes et al. 2013a, 2009; Kälberer et al.

2007]. The frame field's singularities are adopted in this process and define the implied hexahedral mesh's singularity structure.

Unfortunately, this field-guided mapping approach does not guarantee a valid resulting map without flips. While there are heuristics [Lyon et al. 2016] to recover a valid hexahedral mesh even from some invalid integer-grid maps with flips, no general guarantees are available. In the 2D setting, aiming at quadrilateral mesh generation, a stream of recent work has shown ways to reliably generate flip-free maps with prescribed singularities (for instance implied by frame fields) and boundary alignment [Campen et al. 2021, 2019; Campen and Zorin 2017; Gillespie et al. 2021; Shen et al. 2022b]. It is based on phrasing the problem as a constrained metric computation problem; in a specific discrete conformal setting and formulated in per-vertex scale variables, the problem becomes convex and can be solved reliably. Most importantly, these methods employ on-demand mesh refinement (or modification) to ensure feasibility, in the sense that the mesh offers sufficient degrees of freedom to support a valid map, represented in a piecewise-linear manner. Note, however, that field guidance is considered in these works only in the form of adopting the singularities, not in the form of dedicatedly following the field's directions. Generalization of this general approach to the 3D setting is not straightforward; the space of 3D conformal maps too restricted to be useful. The work by [Paillé et al. 2015] may be viewed as a first step: It describes the representation and optimization of discrete metrics on 3D tetrahedral meshes in intrinsic variables (dihedral angles). Using linear constraints in these variables, boundary alignment and singularities can directly be prescribed, for instance those adopted from an optimized 3D frame field. However, the resulting problem in this 3D case is non-convex, and the issue of potentially required mesh refinement is unsolved. While there are known ways to reliably generate flip-free maps in 3D [Campen et al. 2016], they do not support the prescription of arbitrary singularities.

An alternative reliable approach proposed for the 2D setting is based on decomposing the domain into regular pieces based on stream lines of a 2D frame field [Myles et al. 2014]. Also this does not generalize to a 3D stream surface based approach [Kowalski et al. 2016] with similar guarantees.

Finally, the aspect of quantization, as discussed for polycube maps in Sec. 4.7, is relevant for general integer-grid maps as well, in order to ensure the required integer alignment of boundaries, singularities, and other features.

4.9 Hex-Dominant Meshing

Automatic methods for all-hex meshing are only applicable to a subset of all the possible inputs. In contrast, the grid-based approaches (Sec. 4.6) can operate on intricate shapes and guarantee all-hex meshes; unfortunately, they produce inferior quality results. High-quality, feature-aligned all-hex meshes are still elusive, so industry still relies on semi-manual block decomposition, a time-consuming process [Lu et al. 2017].

For this reason, other methods focus on the hexahedral-dominant meshing instead of full-hex, aiming to reach the highest possible proportion of hexahedra. Hexahedral-dominant meshing is a relaxation of the problem to significantly improve robustness at the cost of introducing a small number of generic polyhedra. The generation of hex-dominant meshes boosted the use of those datasets in practical contexts such as FEM [Wicke et al. 2007]. Moreover, the recent advancement in the construction of higher-order bases [Schneider et al. 2019] may foster the adoption of hex-dominant meshes in the mechanical analysis.

The first approach to produce hex-dominant meshes agglomerates neighboring tetrahedrons to assemble hexahedral cells. The problem of finding a globally optimal solution is NP-complete; hence the clustering process is usually driven by local heuristics. Meshkat and colleagues [2000] have proposed the first method following this idea. The clustering process relies on an undirected graph representing tetrahedra and their connectivity. The graph is enriched with particular arcs and labels on nodes to calculate the agglomeration heuristic. Given the initial tetrahedral mesh, the algorithm detects and replaces subgraphs with hexahedral nodes.

The method proposed by Yamakawa et al. [2002] takes as input a general 3D mesh and distributes a set of nodes into the volume by the physical simulation of crystal pattern formation. Then nodes are used to produce a mesh composed of hexes, prisms, and tets, with around 50% hexahedral cells. This method allows controlling element size and primary orientation. Since it does not require a tetrahedral mesh as input, this method is less sensitive to the input discretization than the approach of Meshkat and Talmor [2000].

Vyas and Shimada [2009] proposed a more sophisticated method that starts by generating a volumetric tensor field to specify the anisotropy and directionality of the elements. Then, such a field induces an advancing front process where several hexahedral fronts contribute to cover the entire volume.

Lévy and Liu [2010] generalized Centroidal Voronoi Tessellation [Faber and Gunzburger 1999] for hex-dominant meshing, introducing Lp-Centroidal Voronoi Tessellation. Unlike the standard Voronoi diagram, Lp-CVT favors the formation of cubical cells by using a distance metric that takes into account a predefined background tensor field. The resulting method excels in robustness and controllability.

Despite the considerable advancements in performances, the methods mentioned above cannot obtain a high hex ratio for the general case. The approach proposed by Sokolov et al. [Sokolov et al. 2016] produces higher hex ratios by using a guiding frame field. A sampling process generates a point set organized as a regular grid and locally aligned with the frame field. A constrained Delaunay triangulation of the volumetric samples makes a tetrahedral mesh and is finally clustered into hex elements. They obtain hex-dominant meshes with up to 95% hexahedral cells (although in the worst case they earned less than 30%). Despite the result's quality, this method might produce non-conforming meshes containing configurations where a quadrilateral face is adjacent to two separate triangular faces. This issue has been solved by [Ray et al. 2018].

The approach by Pellerin et al. [2017] explores the space of all possible agglomerations of tets. Then a greedy process selects the configurations to agglomerate tets into hexes. Their approach produces hex-dominant meshes with a 60% ratio of cuboidal elements across all shown examples.

Gao et al. [2017b] directly generate conforming hybrid meshes using polyhedral agglomeration. This method starts from tetrahedral mesh obtained by sampling a guiding frame-field. An iterative

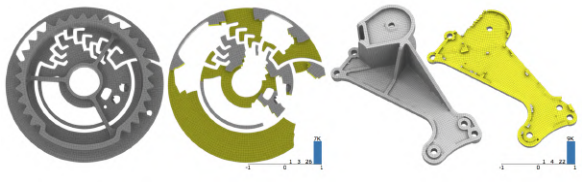


Fig. 20. Some challenging examples of hex-dominant re-meshing using [Gao et al. 2017b].

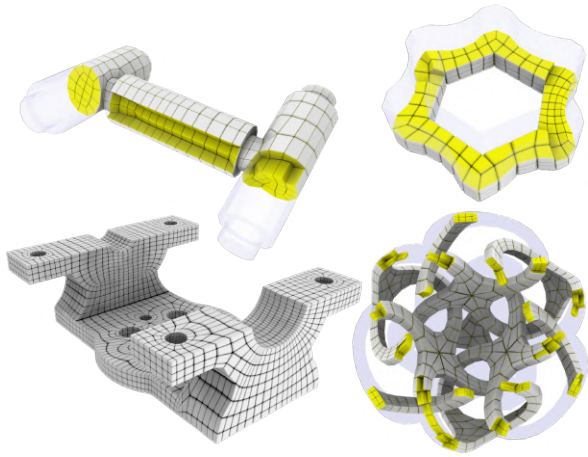


Fig. 21. Some CAD models remeshed by [Livesu et al. 2020]. In this case the meshes are hex-only.

process modifies the connectivity utilizing a set of local operators to compose hexahedral cells. The local operators grant the conformity of the final mesh. While this method excels in robustness (as demonstrated by the complex example shown in Fig. 20), it cannot control the class of created polyhedrons (they might have up to 40 facets in some cases, see Table 1 in [Gao et al. 2017b]).

The recent method proposed by Livesu et al. [2020] produces *strongly hex-dominant meshes*, conforming meshes with less than 2% non-hexahedral cells. In most cases (76% of the models tested), this method derives pure hex-meshes. It mimics manual block decomposition. It extracts first a set of well-distributed loops on the surface following a feature-aligned cross-field. Each loop defines a cutting surface that decomposes the volume into simpler polyhedral blocks. The cutting surfaces are added one by one until the quality requirements of the polyhedral blocks are satisfied. These blocks are finally converted into hex-dominant mesh via midpoint subdivision. As shown in Fig. 21, this method excels in the preservation of sharp features, which are directly incorporated in the output connectivity.

Similarly to LoopyCuts [Livesu et al. 2020], HexDom [Yu et al. 2022] produce a block decomposition where each block is either hex, prism, or tetrahedral cell. This method extends the approach for polycube generation proposed in [Hu and Zhang 2016] based Voronoi tessellation (CVT) to include non-hex elements. The cells are embedded in 3D using a variation of [Yu and Wei 2020]. The segmentation and polycube definition process requires manual work. The methods proposed in [Zhan et al. 2018] and the one used by the commercial package Cubit [Meyers and Tautges 1998] uses advancing front approaches to produce meshes composed of hexahedral and tetrahedral elements. To improve the quality of the final mesh Cubit use some sophisticated cleanup operation based on connectivity editing and geometric measures.

The recent approach proposed by Bukenberger and colleagues [Bukenberger et al. 2021] generates *At-Most-Hexa Meshes*. At-Most-Hexa Meshes are meshes composed mainly of hexahedral elements, where no cell has more than six faces, and no boundary face has more than four sides. Similarly to tetrahedral and hexahedral meshes, the volume of each cell can be defined by trilinear interpolation from its corners. At-Most-Hexa Meshes meshes are generated by extending to the volume the 2D approaches that use Lloyd relaxation with non-euclidean distance measures [Hausner 2001]. Using the L_∞ norm (instead of the simple euclidean distance), the cells emerging from the volumetric Lloyd relaxation process become more cubical, converging to a hex-dominant mesh. Similarly to most of the meshing methods based on Lloyd relaxation, this method is very permissive on the required input. It works on point clouds, triangular meshes and can be guided by an input orientation field if available.

The hex-dominant mesh allows sufficient degrees of freedom to adapt the grid-based methods to conform to sharp features. Trimmed hexahedral meshes are created by intersecting a grid with a closed surface. Non-hexahedral elements emerge along the surface where the surface is not aligned with the edges of the grid. The technique recently proposed by Kim and colleagues [Kim and Kim 2021] extends the trimmed hexahedral methods by creating particular vertices where sharp features intersect with the grid. A feature simplification schema is used when multiple features are concentrated in the same cell.

Another class of methods transforms hex-dominant meshes to increase the number of hexahedral elements in the mesh. The approach proposed in [Yamakawa and Shimada 2003] increases the number of hexahedral and prism elements by applying sequences of local operations that modify the connectivity. Unfortunately, this method can generate non-conformal meshes. Instead, HexHoop [Yamakawa and Shimada 2002] converts a mesh composed of hexahedrons, prism and tetrahedrons into a conformal pure-hex mesh. The conversion process is based on the local application of two particular refinement schemas, called core and caps. Unfortunately, this method tends to insert a high number of irregular vertices deteriorating the regularity of the tessellation.

5 TOPOLOGICAL OPERATORS

Scientific computing often demands to edit a given hexahedral mesh, e.g. to improve the accuracy of a solution with a posterior refinement [Shen et al. 2022a], to generate boundary hex layers for CFD applications [Reberol et al. 2021], to ensure mesh conformity across surface membranes [Staten et al. 2010b] or to constrain the mesh size [Maréchal 2009]. Unlike tetrahedral meshes, where changes

Table 2. Summary of the main properties for each class of hex-meshing algorithms reported in Sec. 4. Some of the columns in this table correspond to items also listed in [Blacker 2000]. As a rough indicator for the extent of (ongoing) research activity, we list the number of overall works and recent (published within the last 5 years) works (referenced in this survey) dealing with each class.

Method	Type	User interaction	Shape class	Feature preserv.	Size contr.	Mesh structure	Element quality	Robustness	Orient. sensitive	Total works	Recent works	Open problems
Advancing front	Direct	Automatic	CAD oriented	Surface features only	No	Unstructured	Good at border, poorer inside	Poor	No	7	0	Improve handling colliding fronts, complex topologies
Dual methods	Both	Automatic, semi-automatic	CAD oriented	Surface features only	No	Unstructured, semi-structured	Good at border, poorer inside	Poor	No	13	2	Robust handling of self-intersecting sheets
Sweeping, decomp.	Both	Semi-automatic	CAD oriented	Surface features only	No	Semi-structured	Good	Good (manual)	No	35	11	Automatic definition of sweepable sub-volumes
Grid based	Indirect	Automatic	Any shape	Yes, (limited valence)	Yes	Severely unstructured	Poor at border, optimal inside	Great (commercial product, demonstrated on many datasets)	Yes	18	3	Feature preservation, mesh size, mapping
Polycube maps	Indirect	Automatic, semi-automatic	Any shape	Yes, (limited valence)	Yes	Valence semi-structured	Good (depends on map)	Good (demonstrated on medium datasets)	Yes	18	8	Polycube topology, mapping, feature preservation
Frame fields	Indirect	Automatic, manual fixing	Any shape	Yes	Yes	Valence semi-structured	Good (depends on map)	Poor	No	17	12	Generation of hexable fields, field aligned mapping
Hex-dominant	Both	Automatic	Any shape	Yes	Yes	Valence semi-structured, hybrid	Often good	Good (demonstrated on medium datasets)	No	20	11	Hybrid elements (topological control, amount, quality)

of the mesh connectivity always have a local footprint, editing the topology of a hexahedral mesh is often a global operation. This makes hexmesh editing significantly more difficult than tetmesh editing. In this section, we revise the most prominent operators for editing the topology of a hexahedral mesh.

5.1 Sheet Operators

As shown in Sec. 2.2, the dual of a hexahedral mesh is a simple arrangement of surfaces. Each surface, i.e., a sheet, corresponds to a layer of hexes in the primal mesh and two surfaces intersect along a chord, which is a column of hexes in the primal mesh. Sheet operators consist in inserting or removing a complete sheet or chord from the mesh. They are used to refine [Ko-Foa Tchon and Camarero 2002; Parrish et al. 2007] or coarsen meshes [Benzley et al. 2005a; Shepherd et al. 2010], to capture analytic features [Merkley et al. 2007], or to make conforming meshes involved in the assembly of parts. Those parts can result from a volume decomposition during a user-assisted meshing process [Borden et al. 2002b; Jankovich et al. 1999] or can correspond to two adjacent models sharing contact surfaces [Staten et al. 2010b].

Using sheet operations to refine a mesh mainly involves inserting sheets, which is quite easy to control if you avoid self-intersecting and self-touching sheets (see Fig. 22). The remaining difficulty is to control the mesh quality, which is connected to the edge valence [Staten and Shimada 2010]. The simplest way of inserting a sheet consists in padding a region of hexahedra by inserting a layer of hexes around it. It is a common post-process to improve meshes obtained with overlay-grid [Maréchal 2009; Qian and Zhang 2010], where a global padding is performed, or Polycube-based [Cherchi et al. 2019a] techniques, where the region to pad is selected in such a way that the mesh quality is optimized (see Sec. 5.4).

Coarsening is much more tricky since some sheets cannot be collapsed without loosing a part of the geometry - resulting in a non-manifold configuration for instance - and one might have to deal with self-intersecting and self-touching sheets, which are much more complex to remove. In [Gao et al. 2017c], such coarsening is performed to simplify the base complex structure. Generating a hexahedral block structure can also be seen as coarsening an existing hexahedral mesh. In [Wang et al. 2017], authors extend the preliminary work of [Kowalski et al. 2012] where a hexahedral mesh, obtained from converting a tetrahedral mesh by splitting each tetrahedron into four hexahedra, is coarsened by removing all non-funda-mental sheets. They extend the greedy approach proposed

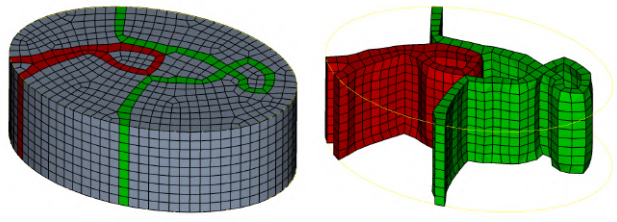


Fig. 22. Examples of self-intersecting sheet (green) and self-touching sheet (red).

in [Kowalski et al. 2012] by providing much more quality control and sheet selection procedures.

In order to make two hexahedral meshes of two geometrical parts sharing a surface conforming, the authors of [Staten et al. 2010b] interleave sheet insertions and collapses in both parts. The locality is controlled by performing chord collapses to avoid the propagation of mesh modifications too far from the interface (see Fig. 23). Chord collapsing is done by taking care of mesh quality, considering edge valences as an appropriate indicator [Staten and Shimada 2010]. In [Chen et al. 2016], the sheet insertion is enhanced to provide more flexibility, in particular, to handle self-intersecting sheets within a local region while assuring the mesh quality. It was successfully applied to mesh matching and mesh boundary optimization.

Eventually, in most recent works, like [Shen et al. 2021], the chord collapse operation is enhanced to avoid generating poor quality elements, and the chord insertion process is described and used for editing the singularities of a hex-mesh while maintaining its connectivity. In [Wang et al. 2018], sheet operations are used in combination with frame fields to improve mesh quality with the ability to handle self-intersecting and self-touching sheets.

5.2 Flipping Operators

Among the many difficulties of hexahedral meshing, there is one that is unexpected, to say the least. The generation of conforming hexahedral meshes of complex 3D domains is definitively a hard problem. Yet, finding hexahedrizations for small quadrangulations of the sphere is also hard.

Existence. Thurston [1993] and Mitchell [1996] have shown independently that a ball bounded by a quadrangulated sphere could be meshed with hexahedra *if and only if* the number of quadrangles on the boundary, n, is even.

Linear Complexity Meshing. Mitchell's construction can necessitate up to $O(n^2)$ hexahedra. In [1999], Eppstein proposed a "semi-constructive" alternative which guarantees the use of $O(n)$ hexahedra. The algorithm of Eppstein extends the quad-mesh in input into a buffer layer of hexahedra. Then it triangulates the inner of the layer with $O(n)$ tetrahedra, applies the midpoint subdivision to split each tetrahedron into four hexahedra, and eventually refines the cubes in the buffer into smaller cubes that consistently meet the previously subdivided tetrahedra. The inserted buffer layer is mandatory to provide much degrees of freedom to topologically and

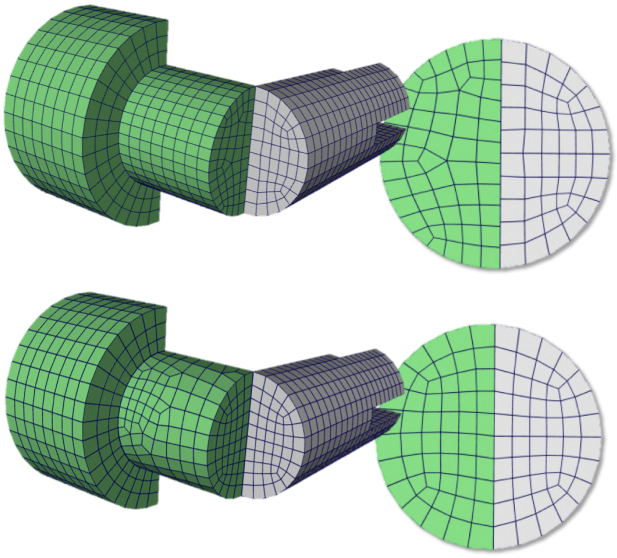


Fig. 23. Two meshes generated using sweeping are not conform along a contact surface (top); Performing sheet operations on the green mesh allows to get a conforming interface (bottom). Image from [Staten et al. 2010b].

geometrically modify the inner mesh. It must be remeshed at the end to ensure mesh conformity. The last stage "only" requires finding a solution of 20 or 22 quadrilaterals buffer cubes. At that point, an explicit solution is required for the buffer cubes. In [2010], Carbonera and Shepherd give the first completely explicit construction of the hexahedrization of the ball. This method, however, requires up to $5396n$ hexahedra. Using [Shepherd et al. 2010], a solution for the buffer cubes has been found by Weill and Ledoux [2019] that involves 76881 hexes! In [2019a], Verheltsel introduced an efficient quad flip-based algorithm that allows finding hexahedral meshes for both the types of buffer cells previously described. Furthermore, as depicted in Fig. 24, it provides geometric realizations with a maximum number of 72 hexahedra, thus proving that it is possible to mesh any ball-shaped domain that is bounded by n quadrangles with a maximum number of $78n$ hexahedra.

Schneiders' Pyramid and Octahedral Spindle. The pyramid of Schneiders is square-based, with 8 additional vertices at the edge midpoints, 5 additional vertices at the face midpoints, and its triangular and quadrangular faces divided respectively into 3 and 4 quadrangular faces. To build the Schneiders' pyramid, we can use the octagonal spindle, or tetragonal trapezoid, and add 4 hexahedra to form the pyramid base. Meshing this pyramid with all-hexahedral elements is a problem introduced by [Schneiders and Bünten 1995] to show a boundary mesh for which no one hexahedral subdivision was identified. A good solution to Schneiders'pyramid is considered as the missing piece to transform a hex-dominant mesh into a full unstructured hex-mesh. To understand this, it is helpful to think at the Schneiders'pyramid as a squared pyramid with one step of midpoint refinement. In this regard, due to the presence of both quadrilateral and triangular faces such an element can be considered as a topological bridge between tetrahedra and hexahedra, with midpoint

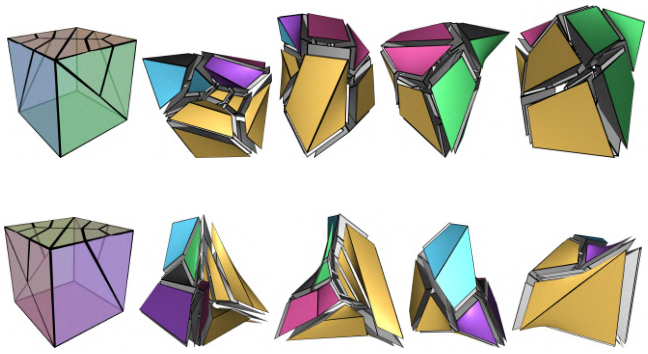


Fig. 24. The set of the hexahedrizations of the buffer cubes that the [Erickson 2014]'s algorithm uses to mesh arbitrary domains. (top) Cells with 20 quadrilaterals are meshed with 37 hexahedra. (bottom) Cells with 22 quadrilaterals are meshed with 40 hexahedra. Elements are colour codeded to show the different sides of the original cubes (top-left and bottom-left). Image from [Verhetsel et al. 2019b].

subdivision being the step that allows to convert all elements into hexahedra. [Shimada and Yamakawa 2002] introduced, in 2002, the hexhoop template family and built a hexahedral subdivision for the pyramid of Schneiders, composed of 118 hexahedra. Later, in 2010, they improved their solution by creating a new hexahedral subdivision of 88 elements [Yamakawa and Shimada 2010]. Recently, in 2018, a hexahedral subdivision of 36 elements was built by finding a set of flipping operations allowing to turn the cube into Schneiders' pyramid, by interpreting each operation as the addition of a new hexahedron [Xiang and Liu 2018]. Verheltsel [2019a] used quad flips to find another solution with 44 hexahedra.

Shellings. In mesh generation, flipping (or swapping) operators convert small cavities of elements into alternative collections of elements having the same boundary. Flips are used extensively in tetrahedral meshing with the aim of *improving the mesh*. The "bistellar flips", the most basic operations, operate on a cavity of 5 vertices produced by removing two or three tetrahedra. Instead, the "edge removal" operation, a more general transformation n-to-m flip, works on a cavity produced by the set of tetrahedra enclosing an edge. Rather than adding more and more operations to the already big set of topological transformations, the "small polyhedron reconnection (SPR)" [Liu et al. 2007] provides an operation that can generalize all the flips. The SPR considers the problem of finding *all the possible triangulations of a cavity* and choosing the best one.

Flipping operators in cubical meshes were introduced by M. Bern, D. Eppstein and J. Erickson in [2002] and are analogous to the flipping operators for simplicial meshes. Those authors prove that each domain that is simply-connected and has an even number of quadrilateral faces also has a pseudo-shelling. A pseudo-shelling is defined as a particular kind of hex-mesh built by adding elements one by one such that the remaining elements always make a ball-shaped domain.

5.3 Atomic Operators

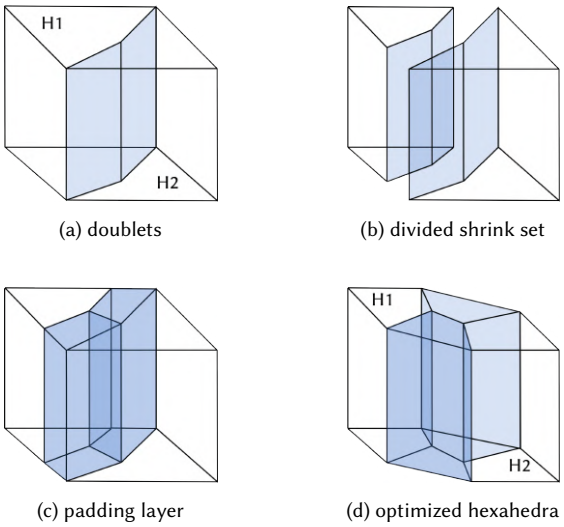
Atomic operators form a set of irreducible local operations which could be composed to described any topological modification [Tautges et al. 2008; Tautges and Knoop 2003]. It consists of three very local atomic operations, which are the atomic pillow, the face shrink and the face open-collapse. An important feature of those operations is that applying just a single atomic operation does not provide a valid hex-mesh. But it has been demonstrated that flipping operations [Tautges et al. 2008] and sheet operations [Ledoux and Shepherd 2010] can be obtained as a sequence of atomic operations. Unfortunately, the completeness of this set of operations is not proved, and they are very difficult to be used for writing meshing algorithms in practice.

For instance, those operators do not capture a parity change in the number of hexahedra. Therefore, an extra operator was presented in [Jurkova et al. 2008], where a Boy's surface is added in the dual mesh representation. The surface of Boy has the interesting property of having a single vertex. Thus, introducing it, in an appropriate way, into the dual mesh representation, the parity of the hexahedra number changes in the primal mesh. In [Jurkova et al. 2008], a sheet diagram is provided, but the primal mesh realization from this insertion is incomplete. Interestingly, utilizing the Carbonera's algorithm [2006] on a single hexahedron, it is possible to perform a parity change. Regardless of the template set of the Carbonera's method, it always replaces a hexahedron with an even number of hex-elements without altering the boundary of the input hex.

5.4 Padding

Sometimes hexahedral meshes (as well as quad-meshes) can contain *doublets*. As described in [Mitchell and Tautges 1995], a doublet is defined as two quad faces sharing two edges, and, in the hex-meshes case, this means that two hexahedra share two faces (Fig. 25a). If doublets occur in a mesh, any kind of geometric embedding of the faces forming the doublet has a low quality, even if we try to optimize it with some smoothing/untangling step. In fact, one of the involved faces will always have an angle of at least π. The local connectivity of the mesh requires a refinement step to remove doublets. The *padding* refinement operation, also known as *pillowing*, refines the mesh structure in order to provide additional elements, and hence degrees of freedom, for existing approaces of mesh optimization (e.g. untanglers). In quad-meshes, this step is trivial. Removing the two shared edges forming a single quadrangular face is sufficient. It is not possible to apply the same for hexahedral meshes because we can not ensure that the hexahedra with doublet faces can be matched in a conformal configuration. In this case, it is required to increase the connectivity of the vertex shared by the two edges that form the doublet.

In [Mitchell and Tautges 1995], the authors propose a pipeline to face the problem in three steps. First of all, a *shrink set* is defined as a set of hexahedra containing one (and only one) of the doublet faces. Then, the set is separated from its boundary by reducing the size of its elements. In this way, an empty space is created (Fig. 25b). Finally, each of the shrink set elements is connected to the boundary through a new layer of hexahedra, filling the previously created empty space (Fig. 25c). After the padding, the original doublet's faces



(a) doublets (b) divided shrink set

(c) padding layer (d) optimized hexahedra

Fig. 25. A summary of the padding pipeline: (a) The hexahedra H1 and H2 share two faces forming two doublets. (b) The shrink set is disconnected to the other elements in the mesh. (c) The elements of the shrink set are linked to the mesh forming the padding layer. (d) The hexahedra involved in the refinement can now be optimized with a smoothing/untangling step.

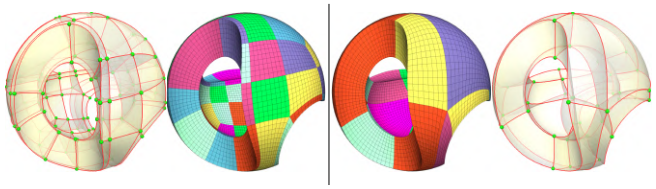


Fig. 26. Without controlling of alignment, the same set of singularities can introduce two hex-meshes having base complexes with different complexity. Image from [Gao et al. 2015].

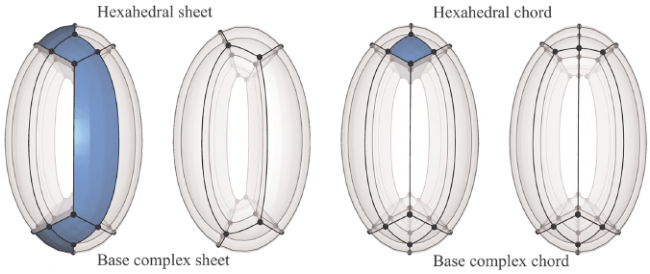


Fig. 27. Removing either a base complex sheet (left) or a chord (right) on the global structure of a hex-mesh monotonically reduce the number of components of the base complex. Image from [Gao et al. 2017c].

are contained in two different hexahedra, doublets are no longer present in the mesh, and the dihedral angles between faces can now be improved (Fig. 25d).

The padding is basically a sheet insertion on a mesh. As described in [Shepherd 2007; Shepherd and Johnson 2008], it is a fundamental step in many applications. Starting from the mesh generation [Gao et al. 2019; Ito et al. 2009] or the generic refinement of hexahedral meshes [Benzley et al. 2005b; Malone 2012; Qian and Zhang 2010; Tchon et al. 2004; Zhang et al. 2013], it becomes an essential ingredient in operations like grafting [Jankovich et al. 1999], mesh cutting [Borden et al. 2002b].

In [Gregson et al. 2011] the padding is identified as a key post-processing step for the hex-meshes obtained from polycubes (see Sec. 4.7). In this mesh category, the surface edges belonging to the polycubes structure can create configurations similar to doublets. The degrees of freedom of the surface elements are then increased with a padding step performed all around the mesh (all the inner volume becomes the shrink set and is separated from the surface). In this way, a geometric optimization step can enhance the quality of the elements placed in the smooth object parts. Notice that, in almost all polycube-based hex-meshing works, the padding operation is applied as a unique hexahedral layer all over the surface. In the same context, in [Cherchi et al. 2019a], a smart and localized padding for this hex-mesh category is proposed. The authors demonstrate that selective padding in sporadic surface areas can significantly improve the whole mesh quality compared to the global padding application.

5.5 Structure Enhancement/Simplification

Hex-meshes with simple structures are preferred for isogeometric analysis [Hughes et al. 2005] since large components allow the fitting of high-order splines without breaking their smoothness so that accurate PDE solving and fast convergence can be achieved. Note that the base complex of a hex-mesh is not only determined by its singularities, but also the connections between them. Therefore, without careful control, the same set of singularities can lead to dramatically different base complexes (Fig. 26). Gao et al. [2015] propose the first solution to reduce the number of components of the base complex by correcting misalignments of singularities. The misalignment correction is achieved by removing hexahedral sheet defined within the base complex. To maintain singularities, specific conditions are posed for choosing the proper hexahedral sheets for removal. After obtaining the hex-mesh with corrected misalignment issue, they employ an extended version of the parameterization-based optimization from quad-meshes [Tarini et al. 2011] to hex-meshes to improve the geometric quality of the hex-meshes. To specifically handle misalignment issue for polycube hex-meshes, [Cherchi et al. 2016] proposes an approach by alternating two steps: (1) computing polycube corner pairs in the integer lattice, and (2) aligning corner pairs through mixed-integer programming.

Robustly producing valid hex-meshes with a simple structure remains to be a challenging task. Gao et al. [2017c] propose a simplification algorithm that can iteratively reduce the structure (i.e., number of components and singularities) complexity of a hex-mesh, while providing several guarantees during the simplification process: (1) topology consistency, (2) inversion-free, (3) the preservation of corner, line, and planar features, and (4) a bounded, user-defined Hausdorff distance from the input surface. The core idea of their approach is to extend the sheet and chord operations on hex element level to the structural level, as illustrated in Fig. 27. The input to this approach can be an arbitrary hex-mesh. Especially, this approach can be paired with octree-based methods discussed in Sec. 4.6 to

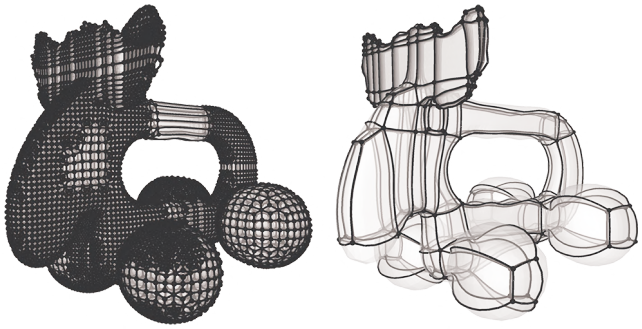


Fig. 28. Turn an octree-based hex-mesh with a highly complex structure into a hex-mesh with a coarse structure. Image from [Gao et al. 2017c].

robustly generate valid (i.e., with no flipped elements) and accurate hexahedral meshes with coarse structures, without user-interactions (Fig. 28). A follow-up work [Xu et al. 2021] is conducted to improve the ranking scheme of the sheets and chords to be removed. The experiments show that, while being more complex, the introduced ranking leads to better simplification results.

Through proving the equivalence between colorable quad-meshes and Strebel differentials on a manifold closed surface, Lei et al [2017] propose to first construct a colorable quad-mesh and then partition the surface into sub-volumes where each of them can be swept to generate a hex-mesh. While the theory is elegant, there are several limitations of the work, prohibiting its adoption for practical applications. For example, the required special user inputs are non trivial, only quad vertices with even valences are allowed, and distortions of the hexahedra could be arbitrarily large.

By adapting the editing operations of singularity pairs for quadrilateral meshes [Peng et al. 2011] to hexahedral meshes, Shen et al. [2021] propose to employ chord collapse and insertion to flexibly control the singularities of a hexahedral mesh. The main limitation of this approach is that chord insertions are not always feasible when the structure of the mesh is complex. The authors demonstrate that the proposed editing operations can be used to clear some connectivity inconsistency issues for sweeping based hex-meshing.

6 GEOMETRIC OPTIMIZATION OF ELEMENT QUALITY

The vast majority of hexahedral meshing algorithms employ a two-step process where the first step generates an initial mesh which is expected to be dominated by well-shaped elements, but often also contains some poorly-shaped and even *inverted*, or negative Jacobian determinant, elements (cf. Sec. 3). This step is typically followed by an optimization step whose goal is to maximize the quality of the mesh elements and specifically to produce an inversion-free mesh, while preserving the meshed domain boundary surface intact. Improvement methods that keep the mesh connectivity fixed while changing only the locations of the mesh vertices, are commonly referred as *geometric optimization*, *smoothing*, or *untangling* methods [Owen 1998; Shepherd and Johnson 2008]. The latter terms are commonly used to describe the methods that specifically focus on reducing, ideally to zero, the number of inverted elements. As noted by Knupp [2001b], for hexahedral meshes there can be more than one definition of inverted elements. He identifies four different scenarios: (a) the integral of the Jacobian determinant over the element is non-positive; (b) the Jacobian determinant is non-positive at any of the Gaussian integration points (used, e.g., in FEM) inside the element (c) the Jacobian determinant is non-positive at any of the element's corners, or (d) the Jacobian determinant is non-positive at some other specific point(s) inside the element. The vast majority of optimization and untangling methods, described below, focus on the scenario (c). The general scenario, seeking for positive Jacobian determinant at *every* point, is tackled by a few methods [Johnen et al. 2017; Marschner et al. 2020] based on optimization formulations attempting to maximize lower bounds of the Jacobian determinant, cf. Sec. 3.1.2. As with mesh generation itself, geometric optimization methods for hex-meshes face some distinctly different challenges from methods for tet-mesh optimization such as [Erten et al. 2009; Freitag Diachin et al. 2006; Kelly et al. 2013; Sastry and Shontz 2014; Scherer et al. 2010], motivating a distinct line of research dedicated to optimizing hex-mesh geometry.

One can easily define an objective function whose global minimum (or maximum) constitutes the best quality mesh possible for a given fixed connectivity (and either hard or soft constraints that hold the surface vertices on the surface of the meshed object). However, essentially all such known objective functions are highly non-linear and do not allow for robust global optimum computation. Thus the core challenge in mesh geometry optimization is to obtain a function and a corresponding optimization method such that the optimum obtained has no inverted elements and maximizes as much as possible the mesh Jacobian or other proxy quality metrics (see Sec. 3).

Consequently, the main difference between the methods is in the optimization strategy used. A few attempts tried to develop generic global optimization strategies which directly optimize the quality across all mesh vertices (Sec. 6.1); however existing methods are not widely used and exhibit inferior performance compared to existing alternatives. Most existing and widely used methods are based on Gauss-Seidel iterations (Sec. 6.1.1), where vertices are relocated one at a time. The advantage of this strategy is that one can explicitly prevent the quality from dropping locally, e.g., preventing the formation of new inverted elements. The drawback is in increased likelihood of converging to a purely local minimum. Recent research investigates different local-global approaches for mesh optimization (Sec. 6.2). This line of research shows great promise, with several methods significantly outperforming prior art. Below we review these three families of methods in more detail.

6.1 Global Optimization

Several authors aim to optimize mesh quality by simultaneously updating all vertex positions, e.g. [Gao and Chen 2016; Yilmaz and Kuzuoglu 2009]; however they only demonstrate results on simple inputs. It is not clear if these approaches can be extended to a more general setting. The use of global non-linear methods for optimizing mesh quality was investigated in depth by Sastry et al [2009] and Wilson [2011] for tetrahedral and hexahedral meshes respectively. Both concluded that global methods that directly optimize hex shape metrics as a function of vertex positions are typically less robust than

the Gauss-Seidel approach, discussed next, and frequently converge to poorer solutions.

6.1.1 Gauss-Seidel Iterations. Older optimization approaches, reviewed by [Frey and George 2008], iteratively relocate interior mesh vertices to some weighted average, or center, of their neighbors, one vertex at a time. Earlier methods used simple geometric averages, e.g., positioning vertices so as to minimize the Laplacian energy around each vertex

$$\min_i ||v_i - 1/N_i \sum_{j \in N(i)} v_j||^2$$

where v_i are mesh vertex positions, N_i is vertex valence, and $N(i)$ are the vertices adjacent to i. This basic framework is often sufficient to produce quality outputs given meshes with simple connectivity and low-detail surface geometry, with quality surface quad-mesh. Recent methods, e.g., [Knupp 2003; Vartziotis and Papadrakakis 2017; Zhang et al. 2009], use more sophisticated local energy formulations that aim to optimize the size of the solid angles at each vertex. For example, Knupp [2003] encode each hex corner geometry via the condition number of a matrix describing the coordinate system at the corner vertex

$$M = (e_0, e_1, e_2)$$

where e_i are the hex edges emanating from the vertex. The smaller the condition number, the better-shaped the hex is locally (corner solid angle closer to 90°). His method uses line-search to move the vertices one at a time so as to minimize the worst or average condition number impacted by the position of this vertex. Such iterative methods are fairly efficient and can be easily parallelized. Unfortunately, applied as-is, such methods do not guarantee an inversion-free output. Specifically, they are often unable to untangle previously inverted element, and when applied as-is are known to frequently introduce new inverted elements near concave features along the boundaries of the meshed domain [Owen 1998]. Several researchers advocate employing these vertex-relocation based methods either pre or post untangling [Knupp 2003; Vartziotis and Papadrakakis 2017]. Specifically, they suggest to constrain each vertex move so as to avoid new inversions, and rely on the untangling methods to resolve all inverted elements. For example, the widely used Mesquite library [Brewer et al. 2003] uses the algorithm of Knupp [2001b] to first untangle a hex-mesh and then improves its quality iteratively moving one vertex at a time using the method of [Knupp 2003]. Constraining all intermediate solutions to remain in the inversion-free space, can produce sub-optimal, local minimum outputs.

Vartziotis and Himpel [2014a] have proposed new formulations of vertex-by-vertex optimization designed for mixed element meshes; while these formulations were successfully demonstrated in 2D space, they have yet to demonstrate those on a hexahedral or hex-dominant input.

Knupp [2001b] proposed an untangling method that focuses solely on correcting inverted hex-elements, while allowing the quality of the non-inverted ones to deteriorate. The energy function he employs is based on the observation that non-negative numbers are equal to their absolute values. Thus requiring the local volume $\alpha = (v_1 - v_0) \cdot (v_2 - v_0) \times (v_3 - v_0)$ at a hex corner v_0 (where $v_i,\ i = 1 \ldots 3$ are the corners adjacent to v_0) to be non-negative can be cast as minimizing the sum

$$\sum_v (|\alpha| - \alpha)$$

over the eight corners of each hexahedron and over all mesh hexahedra. Notably, the optimum of this function can be minimized while the mesh contains zero volume elements. To prevent this configuration, the author suggests minimizing a modified energy

$$\sum_v (|\alpha - \epsilon \bar{V}| - (\alpha - \epsilon \bar{V}))$$

where $\bar{V}$ is the expected average mesh element size (computed as total mesh volume divided by the number of elements), and ϵ is a user defined parameter. This approach demonstrated that this optimized energy is convex as a function of a single vertex position. If a valid solution can be achieved by moving these center vertices, thus using an appropriate convex optimization strategy, this method is guaranteed to untangle all clusters of inverted elements centered around individual vertices. This method fails on many inputs with clusters of connected tangled elements, where only a tandem movement can result in a valid solution.

In some more recent works, e.g. [Ruiz-Gironés et al. 2014, 2015; Wilson 2011; Wilson et al. 2012], iterative local Gauss-Seidel approaches are employed to correct the inverted elements and improve the overall quality of the elements. In these works, the authors use a specifically designed shape metric to avoid convex elements becoming inverted elements and explicitly encouraging untangling of inverted elements. Specifically, they start from a metric introduced by [Knupp 2001a] and modify it to avoid division by zero in the presence of zero volume elements. Given the Jacobian matrix M, the point-wise distortion is defined via the matrix condition number,

$$\nu = \frac{|M|_F^2}{3D(M)^{2/3}}$$

Here $||_F$ is Froebenius norm and $D(\cdot)$ is the determinant. Notably this value goes to infinity as the determinant approaches zero. To avoid instability near zero the authors replace $D(M)$ with $\frac{1}{2}D(M) + \sqrt{D(M)^2 + 4\delta^2}$ where δ is a user specified small value. They propose several different strategies for optimizing the resulting energy. In particular [Ruiz-Gironés et al. 2015] indicates that an approach where each Gauss-Seidel update performs only one step of gradient descent toward the local minimum performs best in terms of output quality. Intuitively, this observation is consistent with avoiding premature convergence to a local minimum. Gauss-Seidel methods such as the ones above are widely used in industry.

6.2 Local-Global Optimization

Multiple recent methods employ local-global approaches for mesh optimization. They conceptually break the mesh into a collection of local, overlapping sub-meshes, and use those in an iterative optimization process. In each iteration, they first optimize each sub-mesh independently, aiming for a solution that is both sufficiently good (inversion free and high quality) and maximally close to the current sub-mesh geometry. They then update the vertex positions globally while striving to maximally retain the geometry of the just computed

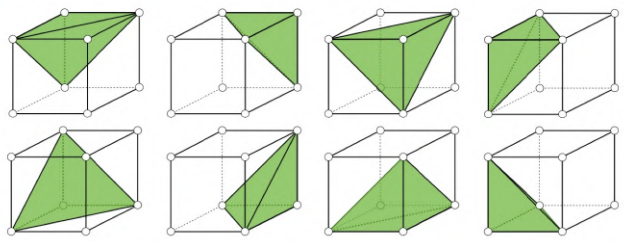


Fig. 29. Corner-based approaches optimize hex shape by considering the set of eight tetrahedra formed by each corner and its three incident edges.

individual local sub-meshes. The two steps are then repeated until no further improvement is possible. The main difference between these approaches is in the choice of the local sub-meshes.

6.2.1 Corner-based. Corner-based approaches consider the eight overlapping simplices formed by the corners of each mesh hexahedron (Fig. 29). These methods were originally proposed for computation and optimization of maps between simplicial complexes e.g. [Aigerman and Lipman 2013; Schüller et al. 2013], but can be applied as-is for hex-mesh optimization, by treating these meshes as consisting of overlapping corner tetrahedra. These methods iterate between optimizing each tet's geometry individually, so as to satisfy a lower bound on quality, and solving for vertex positions that best preserve the resulting individual tet shapes. The global optimization step balances tet shape preservation and preservation of the coordinates of the vertices on the outer surface of the input mesh. As discussed by [Livesu et al. 2015], on many input this approach fails to adequately control the trade-off between boundary surface preservation and quality optimization: holding the boundaries tightly results in poor quality meshes, while relaxing the boundary constraints so as to obtain adequate quality leads to excessive surface drift (Fig. 30).

6.2.2 Hex-based. Marechal [2009] proposes a local-global method that is well suited for grid or octree meshes (with or without padding). At first, a best matching perfect cube is computed for each individual hexahedron. Since each mesh vertex is shared between multiple hexahedra, each vertex receives as target position the average of all the target positions computed for each of its incident elements. Vertices are then carefully moved towards their target position. Geometric fidelity is balanced with per element quality, and surface vertices are allowed to deviate from the nominal surface to avoid introducing flipped elements. To avoid excessive deviation from the input boundaries or corruption of surface features, the method frequently terminates with barely convex meshes, with minuscule minimum scaled Jacobian (≈ 0.01). Further quality improvement using this approach leads to significant boundary drift.

6.2.3 Cone-based. Livesu et al [2015] introduce the notion of cones, sets of hex-mesh corners that surround an *oriented* mesh edge (Fig. 31). They then describe an optimization method that iterates between a local step that optimizes each cone independently to satisfy a minimal quality threshold with minimal changes in cone shape, and a global step that seeks to position all mesh vertices so as to maximally preserve these updated local cone shapes. Both the

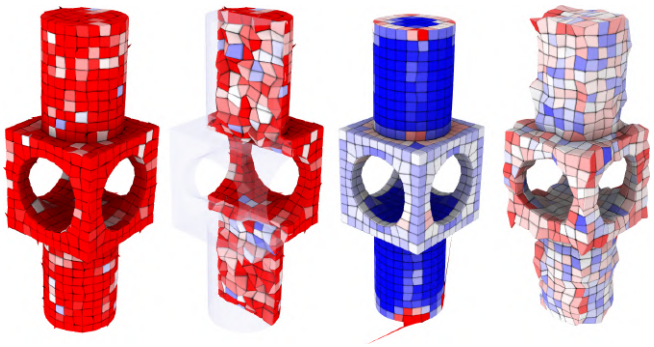


Fig. 30. A degraded hex-mesh obtained by randomly displacing interior vertices (left, first two columns), optimized with a state-of-the-art corner based approach [Aigerman and Lipman 2013]. Hard constraining the surface does not allow to fully untangle the mesh (middle right, see red elements and spikes at the bottom and top). Relaxing it yields a mesh with positive minimum Jacobian, but introduces excessive surface deviation (right). Scaled Jacobian is color coded, from pure red (SJ ≤ 0) to pure blue (SJ = 1).

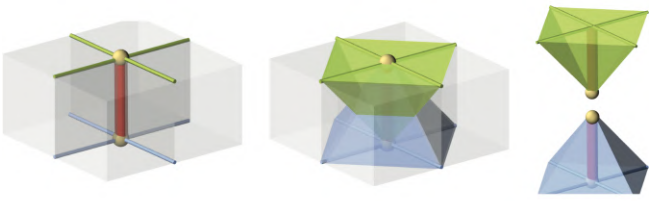


Fig. 31. Cone-based methods cast mesh untangling as the problem of finding a valid axis for pairs of oppositely oriented cones associated to mesh edges. If the axis of each cone stays on the positive half space w.r.t. its base, then the Jacobian at the corners of each element incident to such edge is guaranteed to be strictly positive. Image from [Livesu et al. 2015].

global and the local steps allow feature vertices to move along the underlying features, and surface vertices to move on the object's surface. The method had been shown to provide better balance between surface preservation and quality optimization than prior approaches. Xu et al. [2018] introduce a cone-based two-step optimization strategy whose first step focuses on mesh untangling, potentially at the expense of surface preservation. Their second step then improves surface fidelity and overall mesh quality while preventing the formation of new inverted elements.

6.3 Non-Linear Meshes

The method of Paille et al. [2013] aims to compute low-distortion maps between 3D domains and hexahedral meshes with near-perfect element shape. The method progressively increases the order of the hex elements to improve quality and surface fitting. This approach can be used to optimize the quality of polynomial-basis meshes but is not applicable to standard linear hex-meshes. In particular, while the higher-order elements in the meshes it obtains may be inversion-free and high quality, the underlying linear mesh elements may remain inverted. Most recently, Knupp et al. [2021] proposed a high-order hex-mesh optimization method that targets objects with

no underlying CAD representation but using on the fly computed implicit surface representations. It specifically targets conforming meshes with interior surfaces and is advertised as well suited for computations with dynamically changing geometry. The authors demonstrate the method on a range of simple to medium complexity inputs.

6.4 Simultaneous Geometry and Topology Optimization

Changing mesh geometry and connectivity in parallel can potentially significantly improve the output quality. However, robustly changing the topology of meshes with general connectivity can be challenging due to the global impact of such topological changes (cf. Sec. 5). Thus such research often focused on meshes with near-regular connectivity. For instance Sun et al. [2012] propose an optimization method specific for grid-based meshes that employs a combination of modified Laplacian smoothing and topological operations on the padding layer of the input mesh. The method is demonstrated to work on simple inputs; thus its applicability in the general case remains to be tested. Guided by a frame field generated from an existing hex-mesh, Wang et al. [Wang et al. 2018] propose to identify the hexahedral sheets containing hexahedra with the worst scale Jacobian quality, collapsing the identified sheets, and inserting new sheets with possibly higher quality indicated from the frame field. The insertion of new sheets relies on a stream quad surface extraction which may have robustness issues when the mesh structure is complex.

6.5 Meshes Containing Hybrid Elements

Meshes containing spurious (non-hexahedral) elements demand geometric optimization schemes that are able to improve the quality of arbitrary polyhedral cells. Different from standard finite elements, such as tetrahedra and hexahedra, the literature on general polyhedra is scarce. The manual of the Verdict Library [Stimpson et al. 2007] is a prominent reference for quality metrics of finite elements, and briefly reports only about pyramids, wedges, and knives (Fig. 32), proposing the signed volume as a unique metric, obtained as the sum of the signed volumes of a tetrahedral decomposition of each element. In a recent work, Lobos and colleagues proposed a novel extension of the scaled Jacobian that applies to pyramids and prisms [Lobos et al. 2021]. These elements often occur in hybrid meshes because they are used as *topological bridges* between tetrahedra and hexahedra, or arise when collapsing edges from a regular grid. Nevertheless, some hex-dominant meshing algorithms do not offer any control on the topology of the hybrid elements they create (Sec. 4.9), and may even produce cells for which a tetrahedralization does not exist [Goerigk and Si 2015]. To date, we are not aware of any mesh smoothing or untangling algorithm that can operate on general polyhedral meshes containing elements that do not admit a tetrahedralization.

Restricting to elements for which a tet decomposition exists, mesh optimization algorithms are based around the ideas expressed in [Vartziotis and Himpel 2014b]. The authors start from the consideration that per element volume is not a good metric, because it is scale-dependent, and proposed an alternative metric – called *mean volume* – which is defined on the tetrahedralization of a general polyhedron. The mean volume metric exhibits some desirable properties. In fact, it is scale-independent and is maximized by regular tetrahedra, hexahedra, octahedra, pyramids and prisms. Consequently, following the gradient of the mean volume allows optimizing hybrid meshes made of these elements [Vartziotis and Papadrakakis 2017]. Also, more general elements can be deformed following the gradient, but it remains unclear whether this improves the mesh or not, because of the lack of a canonical reference element. Alternatively, one could tetrahedralize each element and smooth the resulting simplicial mesh, optimizing the shape of each tet. Schemes to convert pyramids, hexahedra and prisms in a globally consistent manner are reported in [Dompierre et al. 1999], and are implemented in open-source tools like CinoLib [2019]. Also in this case, it is not clear to what extent optimizing the tetrahedralization of a hybrid mesh improves the original elements. The topic is indeed under-investigated, and with the proliferation of hex-dominant meshing techniques we expect more and more contributions to be released in the near future.

A parallel line of works is devoted to the study of shape regularity criteria for general polyhedral elements. Shape regularity plays a central role in FEM analysis, as it allows to define precise error estimates on the solution of a PDE, which depends solely on geometric properties of mesh elements. These criteria are well known for triangles, quads, tetrahedra and hexahedra, but the problem hasn't been taken into consideration for general polygons and polyhedra until recently. As of today, shape regularity for arbitrary polygonal and polyhedral elements is relatively strict: concavities are admitted, but elements must have a bounded number of faces and edges, and be star shaped [Lipnikov 2013; Mu et al. 2015]. Some numerical schemes for hybrid meshes (e.g., the Virtual Element Method [Beirão da Veiga et al. 2014]) have empirically shown to be resilient to meshes containing even large amounts of elements that spectacularly violate these criteria, suggesting that more permissive shape regularity criteria could be devised. The problem is still open, and various research groups are working on it. Note that shape regularity criteria are not quality metrics, and involve the assessment of geometric properties that are computationally expensive to evaluate (e.g., being star-shaped) hence they can hardly plugged into mesh optimization schemes. Recent studies are trying to discover new connections between basic geometric properties of mesh elements and the performances of PDE solvers (e.g., approximation error, the condition number of the stiffness matrix), with the ultimate goal to isolate geometric quantities that can be used to drive mesh generation and optimization algorithms in a *PDE-aware* manner [Attene et al. 2021].

7 VISUALIZATION

Researchers involved in mesh generation made extensive use of tools to explore the structure of a volumetric mesh interactively. There are several reasons for the volumetric investigation of hex-meshes:

Evaluation Finite Element analysis often requires the visualization of the result of an experiment, such as the stress distribution or heat propagation.

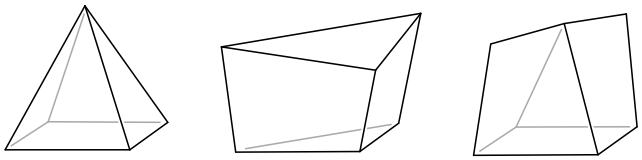


Fig. 32. From left to right: pyramid, wedge, and knife – the only hybrid elements listed in the Verdict Library [Stimpson et al. 2007], a popular reference for the computation of quality metrics of finite elements.

Visualization An interactive visualizer of volumetric meshes is a powerful tool for the visual inspection of a model. Hence, researchers might use it to assess meshing algorithms' performance in terms of element quality or global element arrangement. Interactivity becomes more challenging for high-resolution datasets or meshes with intricate 3D structures.

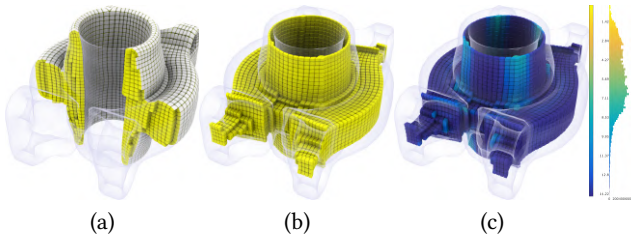
Assessment Secondarily, automated techniques might perform numerical measurements and plotting histograms to assess the quality of a hex model or some 3D field embedded in the elements.

Presentation Finally, if the produced images are of high quality, they constitute a valuable resource for dissemination, e.g., scientific publications.

Visualizing a volumetric dataset in an effective and user-friendly way is a challenging task. The main challenge is to render the internal elements efficiently, even if the external shell occludes them. Volumetric rendering [Balsa Rodríguez et al. 2014] overcomes this limitation by integrating the inner field along a particular view direction. However, besides their practical use in the exploration of biomedical data or FEM, they cannot be effectively used to visualize the mesh's connectivity and the quality of its elements.

An alternative trend enables an interactive user-guided visual exploration of hex-meshes directly via cell filtering or using transparency to reveal mesh internal structure. This approach allows for a detailed analysis of the mesh structure, isolating weak points, or degenerate elements. Some basic library offers a set of essential tools to filter and visualize the elements selectively [Livesu 2019]. Other advanced geometry processing [Levy 2022b] or mesh generation [Geuzaine and Remacle 2009; Zheng et al. 1995] tools offer necessary instruments to examine the internal cells, such as sweeping planes. Similarly, Paraview [Ayachit 2015] and [ANSYS 2022] provide some methods for the visualization and exploration of volumetric datasets, including additional tools to plot and elaborate statistics on volumetric fields embedded in the elements. Besides their use in most application contexts, none of the tools mentioned above is tailored to hexahedral meshes. A different generation of software like Hexalab [Bracci et al. 2019] or the method presented in [Xu and Chen 2018] have been designed explicitly for hexahedral mesh visualization.

Hexalab offers a set of interactively controlled tools to reveal the internal structure of the mesh. The user can either use an interactively controlled sweeping plane (see Fig. 33 a) or peel the object surface layer-by-layer from the outside (see Fig. 33 b). Even if removed from the visualization, the outer surface can be visualized with some transparency effect. Hexalab also provides a high-quality

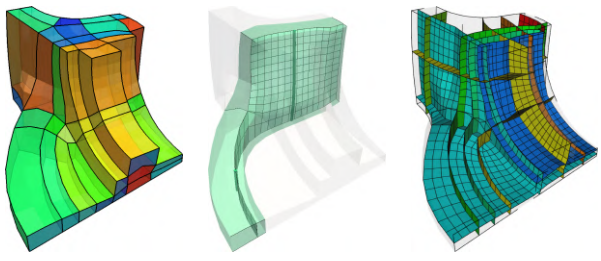


(a) (b) (c)

Fig. 33. The interactive visualization tools offered by Hexalab: Internal exploration using a sweeping plane (a) or the peeling tool (b); Coloring elements by their quality and the resulting histogram (c).

rendering, including non-photorealistic effects on the GPU, like ambient occlusion, to enhance the internal structure and the arrangement of the elements and better communicate the shape of the cells. It also implements all the quality measures in [Gao et al. 2017a], offering automated techniques to numerical assess the quality of a mesh and plot histograms for the inspected model (see Fig. 33 c). Hexalab is also an easily accessible portal online repository of hex-meshes, including a variety of results from various state-of-the-art techniques. This characteristic makes this tool an excellent platform to compare the performance of the different meshing techniques.

While Hexalab provides essential tools to visualize the global connectivity, such as the location and valence of singularities, the approach of [Xu and Chen 2018] provides a sophisticated method for the exploration and the visualization of the volumetric structure. This method exploits the connectivity between base complexes composing the hex-mesh. The structure and connectivity of base complexes provide an excellent overview of the hexahedral elements' underlying flow. The base complexes are groups of cells delimited by the sheets emanating from singularities (Fig. 34 a). Adjacent base complexes following the same frame field directions compose dual-sheets (Fig. 34 b). A global optimization process uses the connectivity between the dual-sheets to select the most significant ones and finally provide an efficient instrument to navigate the high-level overview of the underlying structure (Fig. 34 c). The recent approach proposed by Neuhauser and colleagues [Neuhauser et al. 2021] makes use of GPU shader functionality to generate an advanced volumetric rendering that focuses on a subset of elements.



(a) (b) (c)

Fig. 34. The pre-processing pipeline of [Xu and Chen 2018]: (a) Extracting the base complex; (b) One Dual sheet layer; (c) The final visualization.

This strategy allows, for example, to visualize poorly shaped elements or the elements surrounding a singularity and, at the same time, gradually blend the visualization with the surrounding structure.

Most of the current visualization tools specialize in exploring the mesh connectivity and assessing elements' quality and arrangement. However, most of the tools presented in this section show their limits when employed in an actual industrial application.

The volumetric visualization systems discussed in this section do not scale directly to massive datasets composed of millions of elements. As the opposite, the industrial systems (such as Paraview [Ayachit 2015] and Ansys [ANSYS 2022]) allow for the visualization of meshes composed of millions of elements. However, the rendering quality provided by such commercial packages is usually not as informative as the one provided by the recent tools proposed in academia. Because meshes composed of millions of elements are the standard in several FEM contexts, open-source tools like Hexalab must bridge this gap to have a chance of significant impact in industry.

When a dataset becomes massive, current exploration tools based on transparency, ambient occlusion, or slicing planes might become inadequate to ensure full visual access to the volume. We believe that future visualization tools could overcome these limitations by exploiting current VR developments and possibly interactive gesture tracking.

Finally, most application scenarios require the visualization of complex fields defined over each volume element (such as stress or tensor fields), exploring their variation, and doing some statistics. While Paraview [Ayachit 2015] offers already some advanced tools to this scope, renderings are still not adequate to the state-of-the-art techniques. The modern GPU architecture that supports real-time ray-tracing can trace a new path for the advanced volumetric rendering of hexahedral meshes with complex embedded fields.

8 CURRENT TRENDS AND FUTURE PERSPECTIVES

In this section, we report on the status of hex-meshing as a whole and its future perspectives. Specifically, in Sec. 8.1 we discuss open theoretical problems that are relevant for hex-mesh generation algorithms. In Sec. 8.2 we report on algorithmic issues of existing approaches. Unlike the previous section, in this case, the theoretical formulation of the problem is clear, but current solutions are unsatisfactory, mainly for the intrinsic complexity of the problem tackled. In Sec. 8.3 we indicate various activities that, even though they do not directly advance the state of the art, may foster new research and facilitate the development of better techniques.

8.1 Theoretical Challenges

Characterization and Synthesis of Hex-Meshable Frame Fields. The fully automatic applicability of frame field based techniques (Sec. 4.8) is currently limited to rather simple shapes. User intervention to cure imperfections in the guiding field are required to handle general shapes. As reported in the dedicated section, the space of frame fields is topologically larger than the space of hexahedral meshes. This raises two fundamental theoretical questions:

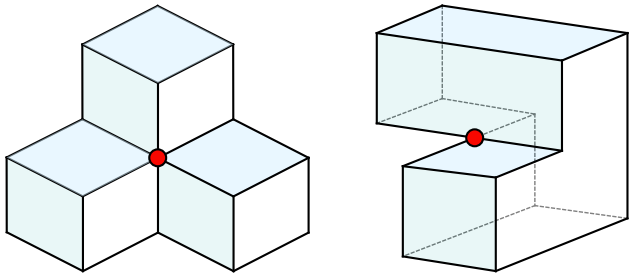


Fig. 35. Two polycubes having non 3-connected graphs. The red nodes at the left is 6-connected; the one at the right is 4-connected. These orthogonal polyhedra are not included in the graph characterizations provided in [Eppstein and Mumford 2010] and [Zhao et al. 2019].

(1) Given a frame field, how to algorithmically verify whether a valid hexahedral mesh of identical topology exists?
(2) How to restrict frame field synthesis algorithms to operate in the space of fields that admit a hexahedralization?

As a variant of the latter question, one may consider a restriction to a subspace of that space – which raises the question what subspace is simple enough to enable an efficient restriction, yet large enough to enable high-quality hexahedralizations (with proper boundary and feature alignment, sizing, adequate levels of regularity, etc.). Unfortunately, both questions are still unanswered and demand further research to understand these geometrical entities at a deeper level. For the characterization of hex-meshable fields, in [Liu et al. 2018] the authors enumerate all local conditions for hex-meshes having singular edges with valence 3 and 5. Also they report a global condition which is a discrete version of the Poincaré-Hopf index formula. While their characterization can be easily extended to a broader set of singular edges, as reported by the authors, the global condition is necessary but not sufficient, meaning that there may still exist fields that obey all these criteria but do not admit a valid mesh (e.g., due to limit cycles [Viertel et al. 2016], or other global inconsistencies [Sokolov and Ray 2015]). The second question cannot be answered at this point either, due to a limited understanding and a lack of simple sufficient and necessary conditions characterizing hex-meshable fields. To sidestep this issue, many authors start by computing an unconstrained frame field, then cure it with manual fixing [Liu et al. 2018] or adopting local heuristics [Jiang et al. 2014; Li et al. 2012; Reberol et al. 2019; Viertel et al. 2016], and then use the corrected singular graph to bootstrap methods such as [Corman and Crane 2019; Liu et al. 2018] which produce a smooth field that conforms to a prescribed singularity structure. Not only is this a cumbersome pipeline, but considering the inability to precisely state whether a field is hex-meshable or not, failure is still possible, even for semi-automatic methods that put the user in the loop.

Characterization of Polycube Surface Structures. Various existing polycube methods (Sec. 4.7) exploit a graph characterization of orthogonal polyhedra [Eppstein and Mumford 2010] in attempts to ensure that the domain structures they generate actually correspond to polycubes. This characterization, however, is unnecessarily limiting. For instance, it is restricted to 3-connected graphs, meaning

that each polycube corner will have exactly three incident polycube edges. As a result, valid polycubes such as the ones shown in Fig. 35 are not considered valid. An even more restrictive aspect is the limitation to genus zero polycubes, though this can be alleviated to an extent [Zhao et al. 2019]. Extending this characterization to 4- and 6-connected graphs, as in Fig. 35, has high practical importance, especially for the purpose of meshing CAD shapes, where more than three sharp features not rarely meet at one point and the inability to reproduce this layout in the generated polycube would inevitably result in a lack of feature preservation and – most likely – in severe and unnecessary geometric distortion. At the same time, as pointed out in Sec. 4.7, the purely graph-based approach is not sufficient to recognize structural issues, as it ignores the fact that in the polycube context one operates with a surface-embedded graph structure. The approach of [Sokolov 2016] goes further in this regard, but inherently does not support abstract, generalized, chart-based polycubes. A fully satisfactory, necessary and sufficient criterion and associated method to (i) detect (or even right away prevent) invalidities, e.g., in surface labelings and (ii) perturb them into the nearest valid solution is still elusive, and this remains an open problem not only for practitioners in mesh generation.

8.2 Algorithmic Challenges

Volume Mappings. Many indirect methods that operate on a supporting tetrahedral mesh rely on a mapping between the input object and a parametric space embedding the mesh connectivity. A common desideratum to promote element validity is that this mapping is (globally or locally) injective. The generation of injective volumetric maps is a broad topic that finds application in many fields. While there are reliable approaches for the computation of such maps [Campen et al. 2016], they are not versatile enough for hex-mesh generation. In recent years some more flexible methods with improved success rates have been introduced [Du et al. 2020; Garanzha et al. 2021] but none of them can actually guarantee an injective result and failure cases are easily encountered in the hex-meshing context. Besides injectivity, it is also important that the map has low geometric distortion. Elements should preserve their good shape through the map so that a regular grid in parametric space translates into a well-shaped uniform hex-mesh of the target object. To this end, current relatively robust methods such as [Du et al. 2020] fall short, because they are focused more on the validity of the solution than on the distortion of the map, and may therefore produce valid meshes that are unusable in practice. Recent literature has shown that adaptively sampling the parametric space can be used to counterbalance map distortion, even in extreme cases (Fig. 18). Nevertheless, devising new algorithms that provide guarantees of robustness and minimize geometric distortion (possibly editing mesh connectivity) will be highly beneficial for many hex-meshing algorithms.

Volume mappings become even more complex when integer constraints are added to the formulation, leading to a mixed-integer problem. These problems typically arise in frame-field based methods (Sec. 4.8) due to the presence of integer transition functions and integer alignment conditions, but may also arise in certain polycube methods (Sec. 4.7), to ensure that input features and polycube edges map to integer isolines in parametric space. As the resulting mixed-integer problems are very hard to solve to the optimum [Bommes et al. 2010], heuristics are commonplace, e.g. based on rounding [Jiang et al. 2014; Li et al. 2012; Nieser et al. 2011] or reduction to linearly constrained integer programs [Brückler et al. 2022a,b; Cherchi et al. 2016]. Yielding a map that is not only of low distortion on average, but strictly locally injective can be even more challenging in this integer-constrained setting.

Regardless of the presence of integer constraints, the generation of the hexahedral connectivity is a discrete sampling, hence the fulfillment of map injectivity (or lack thereof) does not guarantee a success or a failure. A lucky enough sampling of an invalid map may yield well-shaped hexahedra as much as a coarse enough sampling of an injective map may produce inverted elements with negative Jacobian determinant. Nevertheless, valid maps with no inverted elements and low geometric distortion are a good proxy for the generation of well-shaped hexahedral elements, and this is what existing methods strive for. Moreover, from a theoretical standpoint, injectivity guarantees the existence of a (dense enough) sampling where the so generated hexahedra are valid, even though the resulting sampling density may be so high to become impractical for real applications.

Feature Transfer. By their very definition, direct methods are guaranteed to conform to the input geometry and all its features. Conversely, indirect methods can only produce an approximation of the target shape. Many indirect methods insert all (or a part) of the input features after the mesh generation stage, resolving a *feature transfer* problem. This happens for all grid-based methods, but may also happen for methods based on domain decomposition or polycube methods (e.g., to restore features that do not map to polycube edges). Feature transfer is primarily a topological problem because feature lines must be assigned to chains of edges in the hex-mesh, ensuring that no spurious overlaps or intersections are introduced. Known methods operate heuristically, inserting one feature curve at a time along some pre-computed sequence (e.g., sorting features by their length). While the first insertion is free to occupy any mesh edge, the subsequent ones are constrained by previous insertions, clearly designing a combinatorial problem with exponential complexity. Recent literature has shown that a greedy processing sequence may lead to catastrophic results (see, e.g. Figs. 1,2,11 in [Born et al. 2021]). Besides the intrinsic complexity of the general problem, the sparse hex-mesh connectivity and impossibility to apply local refinement to increase the valence of a vertex (e.g. to accommodate more incoming arcs) makes this problem much more challenging on structured meshes than on unstructured ones. Current methods such as [Gao et al. 2019] are limited to simply discarding a feature line if a non-intersecting chain of edges can be computed, and insert it otherwise, regardless of its geometric deviation from the target curve. The use of more sophisticated heuristics such as [Born et al. 2021] may significantly increase the number of features successfully transferred and also help reduce the geometric distortion due to a wiser choice of the feature edges. Alternative methods tailored to operate on the connectivity of hexahedral meshes may also be developed, and coupled with (as local as possible) hex-mesh refinement techniques to resolve intricate configurations. Considering the limitations of

the hex connectivity application-aware de-featuring may also play an important role [Buffa et al. 2022].

Volumetric Modeling. The prominent idea at the basis of Isogeometric Analysis (IGA) is to establish a unified geometric representation for both modeling and simulation, thus avoiding the need to iteratively convert from one representation to the other throughout the product development cycle [Hughes et al. 2005]. Differently from Computer-Aided Design (CAD), which is historically concerned with boundary representations (B-Rep), numerical simulation often necessitates an explicit volumetric representation of the product (V-Rep), which typically comes in the form of a finite tetrahedral or hexahedral mesh. Therefore, the fulfillment of the IGA principles passes through the adoption – also for the design part – of V-Reps. Not only IGA, but also modern manufacturing techniques call for this transition: composite materials and internal microstructures do not scale well with B-Reps, and would benefit from an explicit volumetric representation. Volumetric modeling that operates with tensor products [Antolin et al. 2019b,a; Massarwi and Elber 2016] has close analogies with the definition of structured hexahedral meshes that endow a coarse block decomposition. To this end, advances in this field will go hand in hand with advances in user interactive tools for the generation of semi-structured hexahedral meshes. A few proposals already exist in the literature, but the topic is quite new and under-investigated.

User Interaction. Since no existing hex-meshing method combines robustness, quality, and generality in a fully satisfactory way, manual or semi-manual hexahedral mesh generation is still a prominent approach in industry [Lu et al. 2017]. Professional software such as [ANSYS 2022; CUBIT 2022] and many others are based on interactive pipelines where the user provides a high-level understanding of the object, and is required to instruct the program on how the shape can be split into simpler parts. If and when parts become sufficiently simple, direct methods such as sweeping and advancing front are launched to complete the discretization. Parts that are not sufficiently simple will remain empty, and the user is required to modify the current partitioning or split the non-meshed elements into simpler sub-components. This process is extremely tedious, and requires the user to "understand" (and overcome) the limitations of direct meshing approaches, in order to provide a decomposition that nicely combines the necessity to keep the number of parts low and at the same time simple enough to be processed separately in an automatic fashion [CoreForm 2022b]. Since these tools follow a divide-and-conquer approach, direct hexmeshing techniques are preferred to indirect ones, because they ensure that the meshes of all sub components will be conforming. In recent years, academic literature has started to explore the possibility to couple user guidance with indirect approaches that operate on a supporting tetrahedral mesh [Li et al. 2021; Takayama 2019; Yu et al. 2022; Yu and Wei 2020]. These methods are not based on the typical divide-and-conquer paradigm, and their ability to scale on complex shapes it yet to be demonstrated.
The usefulness of interactive approaches is twofold: from the perspective of mesh users, they allow to hex-mesh objects that would not be possible to produce automatically. From the perspective of practitioners mesh generation, these interactive pipelines often permit to spot the weak parts of the pipeline, isolate corner cases, and interactively explore alternative solutions. To this end, these tools may be highly important for the development of better (i.e., more robust) fully automatic methods.

Hex-Mesh Booleans. In medicine, the simulation of human organs often relies on templated hexahedral or hex-dominant meshes that well capture biological structures such as separation tissues or the alignment of muscular fibers, effectively reproducing their activation [Buchaillard et al. 2009; Gérard et al. 2006; Rohan et al. 2017; Schonning et al. 2009; Takhounts et al. 2008]. Considering the important information encoded in the connectivity of these meshes, when simulating complex body dynamics that involve multiple organs it becomes important to create composite simulation domains that preserve as much as possible the connectivity of each original mesh. Blending multiple meshes into a single one is a widely studied problem in the literature, especially for the case of unstructured meshes composed of triangles or tetrahedra [Cherchi et al. 2020; Diazzi and Attene 2021; Hu et al. 2018; Zhou et al. 2016]. For structured meshes made of quads or hexahedra, the problem is more complex because the necessary changes of the local connectivity have a global footprint. Recently, in [Nuvoli et al. 2019] the authors introduced a method to blend quadrilateral meshes with minimal topological impact. Extending this idea to volumetric hex-meshes remains an appealing avenue for future research with a significative potential impact for bio-medical applications.

Scalability. With the recent advancement of additive manufacturing and topology optimization strategies, mechanical shapes are rapidly growing in complexity. Consequently, hex-meshing methods need to comply with this trend by providing the ability to process large datasets at a reasonable computational overhead. Scalability has not been a central point for most of the proposed methods, but it will increase in importance in the years to come.

8.3 Practical Challenges

PDE-aware Volume Meshing. As discussed in Sec. 4.2, a deeper fundamental understanding of the connection between a hex-mesh and the final application is required. Most of the hex-meshing methods strive to ensure every element to have a positive Jacobian determinant. While this is already hardly achievable reliably with most of the proposed methods, even a positive Jacobian determinant throughout the entire mesh only avoids the presence of degenerate elements. Still, it does not ensure the mesh fits with the target application. The precise connection between a hex-mesh and its final application is usually elusive. In Sec. 3, we presented several quality measures for individual hexes. Still, even for Finite Element Analysis, it is not clear how those metrics impact the accuracy of the simulation in detail. Other applications might prefer the alignment of the elements to a particular vector field over their individual quality. Recent literature has started to investigate the correlation between geometric quality and the accuracy of numerical solvers at a deeper level. A whole line of research is devoted to the evaluation of the

Virtual Element Methods for polygonal and polyhedral meshes [Attene et al. 2021; Cabiddu et al. 2021; Sorgente et al. 2022, 2021]. More related to the topic of this survey is the study published in [Gao et al. 2017a], who conducted a statistical correlation analysis between hexahedral meshes obtained with various techniques, and the resolution of a few representative PDEs. While it remains difficult to design mesh generation algorithms that can address geometric quality criteria at the mesh generation stage, a few exceptions exist. For example, the VoroCrust algorithm [Abdelkader et al. 2020] is designed to intrinsically satisfy the orthogonality criterion required by CFD solvers, obtained with a wise positioning of the Voronoi seeds that fully avoids the necessity to cut (or *clip*) Voronoi cells. It would be interesting to investigate similar ideas to obtain a tighter coupling of mesh generation and its downstream applications.

Tets vs Hexes. Meshes made of hexahedral elements were traditionally considered superior to tetrahedral meshes, both in terms of performance and accuracy. Tuchinsky and Clark observed that since a hexahedral mesh can cover the same volume of a tetrahedral mesh with roughly one-quarter of the elements, there is a 75% saving in terms of computational cost [Tuchinsky and Clark 1997]. This estimate is based on the assumption that "analysis setup and pre-processing requires the same time for hex- and tet-based work", which does not reflect the current state of mesh generation because creating and processing tetrahedral meshes is significantly easier and more robust than creating hexahedral ones [Diazzi and Attene 2021; Hu et al. 2020, 2018]. Regarding accuracy, it seems to be well understood and established that linear tetrahedra are to be avoided because they introduce artificial stiffness in the problem (i.e., they "lock") [Wang et al. 2004], whereas linear hexahedral elements do not introduce such artifact. Typically locking depends on the number of degrees of freedom [Frâncu et al. 2021] and disappears when higher order bases are used [Wang et al. 2004]. This makes hexahedral meshes particularly suited for problems where linear elements are used, such as in the interactive simulation of hyper-elastic and plastic phenomena (e.g. in surgical simulation [Gao and Peters 2021]) and in fast transient dynamic phenomena that employ explicit time integration (e.g. crash and impact simulation) [Gravouil et al. 2009] because higher-order basis functions would necessarily demand a reduction of the time step to achieve numerical stability, according to the Courant–Friedrichs–Lewy condition [Courant et al. 1967; Weber et al. 2021]. In recent years some scientists have questioned the superiority of hexahedral elements over tetrahedra and advocate the use of tetmeshes with quadratic basis functions as general purpose simulation domains [Schneider et al. 2022]. The topic is somewhat orthogonal to this survey, which focuses only on hex-mesh generation aspects. Whether it is for their (uncertain) superiority or because of the presence of highly trusted legacy code that runs only on hexahedral grids, the interest for hexahedral meshes is still high both in industry and in academia. This is witnessed by the growing number of scientific articles published in recent years [Beaufort et al. 2022], by the central role that hexahedral grids occupy in industrial and commercial software, and ultimately by the interest that the industry manifests for each advancement in hex-mesh generation. It is worth noting that, despite their importance, tetrahedra and hexahedra cover just a fraction of the possible simulation domains. General polyhedral meshes made of Voronoi [Lévy 2022] or cut [Tao et al. 2019] cells are a valid alternative and are particularly appreciated in Lagrangian setups, where the mesh evolves over time and must be quickly generated to track features in a simulated domain. For obvious reasons, this survey does not cover this body of literature.

File Formats. Many algorithms for hex- and hex-dominant mesh generation necessitate the ability to process general polyhedral meshes, either at the intermediate steps of the pipeline [Gao et al. 2019; Livesu et al. 2021; Maréchal 2009] or directly in the output mesh [Gao et al. 2017b; Livesu et al. 2020]. In general, these methods put no constraints on the topology of each cell, which can therefore contain any amount of vertices, edges and faces. While data structures capable of handling these entities exist (Sec. 9), we are not aware of any widely accepted file format that allows to save and load output hexahedral dominant meshes or intermediate steps of hex-meshing pipelines. To our knowledge, popular tools such as VTK [2022] only support file IO of *canonical* finite elements, such as tetrahedra, hexahedra, pyramids and wedges, while methods that produce meshes with more complex elements all rely on ad-hoc formats that were released alongside the algorithms themselves [Gao et al. 2017b; Livesu et al. 2020], thus limiting the possibility to exchange material and ultimately triggering a proliferation of alternative proposals. Considering the growing importance of hex-dominant meshes, it would be important to define a file format for general polyhedral meshes, so that groups working in the field can store and release their data in a way that is intelligible by the other groups, and that can be easily supported by third party software such as [Bracci et al. 2019] (e.g., for visualization, comparison, and analysis).

Datasets & Benchmarks. In recent years the computer graphics community has released multiple databases that have been extremely useful for practitioners in the field, raising the bar for new algorithms in terms of scalability and ability to handle a variety of inputs with different complexity, from easy ones to highly challenging. To make a practical example, the Thingi10K [Zhou and Jacobson 2016] dataset has quickly become a popular means to empirically validate the robustness of surface mesh generation and processing algorithms [Hu et al. 2018; Pietroni et al. 2021] and some of its models are so pathological that being able to process them is an achievement by itself, with authors reporting both running times and memory consumption (see e.g. Fig. 17 in [Hu et al. 2020] and Fig.1 in [Cherchi et al. 2020]). To this end, new methods for hexahedral and hex-dominant meshing can greatly benefit from the release of similar databases. The Hexalab project [Bracci et al. 2019] collects output data from the most prominent mesh generation methods in the literature, but it is not meant to be a validation database for novel methods. A few contributions in this direction have been proposed very recently: [Ledoux 2022; Reberol et al. 2019] propose input CAD models, while [Beaufort et al. 2022] offers input tetrahedral meshes with tagged feature entities. Specifically, hard constraints on the preservation of feature curves and (boundary and interior) surfaces can be very challenging for meshing algorithms.

Beyond PDEs: Novel Applications. The numerical resolution of Partial Differential Equations (PDEs) is by far the most prominent

application of volumetric meshes in general, and of hexahedral meshing in particular. Nevertheless, in recent years both meshes of this kind and techniques that were originally developed in the field have been used in alternative applications, such as topology optimization and advanced manufacturing [Arora et al. 2019; Stutz et al. 2022; Wu et al. 2021]. To this end, current themes in automatic hex-mesh generation are beneficial not only for the numerical resolution of PDEs, but may also reach a broader audience.

9 AVAILABLE RESOURCES

Besides various professional and semi-professional tools such as VTK [Schroeder et al. 1998] and its front end ParaView [Ayachit 2015], Cubit [CUBIT 2022], MeshGems [Distene SAS 2022], Gmsh [Geuzaine and Remacle 2009], CoreForm [CoreForm 2022a], CGAL [Fabri and Pion 2009] and many others, over the years academics have released both data and a variety of open-source tools to aid not only their research, but also the activities of other practitioners in the field. This section summarizes the most prominent available resources for hexahedral and hex-dominant meshing. Note that the list of authors releasing their data, code and toolkits is in constant evolution.

Datasets. In Tab. 3 we list datasets released by authors of the methods surveyed in Sec. 4. This includes in particular sets of example hexahedral meshes generated by these various methods, but also hex-dominant meshes (e.g., [Gao et al. 2017b]) as well as challenging input models (e.g., [Beaufort et al. 2022; Ledoux 2022; Reberol et al. 2019]). The HexaLab project [Bracci et al. 2019] is a unified portal to visualize hexahedral meshes directly in a web browser as well as to download them. It collects meshes produced with the most recent techniques in the field, with a focus on pure hexahedral meshes, and includes most of the output data listed in Tab. 3.

Algorithms. In recent years, more and more authors are releasing their source code, either through activities for the reproducibility of scientific experiments, such as [Bonneel et al. 2020; GRSI 2022], or simply by publishing their code on Github or similar portals. In Tab. 4 we report on all the implementation of algorithms surveyed in this article, both in the form of source code or pre-compiled binaries.

Toolkits. While there exist countless open source libraries for the processing of surface (e.g. triangular) meshes, the number of tools that offer data structures for volume meshes is scarce. Besides, most of these tools are dedicated to tetrahedral meshes only. They do not support alternative cells, such as hexahedra or general polyhedral elements that may arise at the intermediate steps of the meshing pipeline [Livesu et al. 2020; Maréchal 2009; Pitzalis et al. 2021], or in hex-dominant methods. In Tab. 5 we report on the most prominent existing software tools for volume mesh processing, summarizing their main features w.r.t. the scope of this survey.

Table 3. List of input/output datasets for the hex and hex-dominant methods surveyed in this article available at the time of writing. Meshes that are included also in the HexaLab database [Bracci et al. 2019] are flagged accordingly.

Method	Data Available	File Formats	On Hexalab	URL
[Gregson et al. 2011]	output hex-meshes	.MESH	yes	zip
[Li et al. 2012]	output hex-meshes	.VTK	yes	link
[Livesu et al. 2013]	output hex-meshes	.VTU .MESH	yes	webpage
[Huang et al. 2014]	input tet-meshes polycube maps output hex-meshes	.VTK	yes	zip
[Livesu et al. 2015]	input hex-mesesh output hex-meshes	.MESH	yes	zip
[Fang et al. 2016]	input tet-meshes output hex-mesh	.VTK	yes	zip
[Fu et al. 2016]	input tet-meshes polycube maps output hex-meshes	.VTK	yes	link link
[Cherchi et al. 2016]	input polycubes (hex) output polycubes (hex) intput hex-meshes output hex-meshes	.MESH	yes	zip
[Livesu et al. 2016]	input surface meshes input curve-skeletons output hex-meshes	.OBJ .SKEL .MESH	yes	zip
[Gao et al. 2016]	input surface meshes output hex-meshes	.OFF .MESH	yes	zip
[Wu et al. 2017]	output hex-mesh	.MESH	yes	–
[Livesu et al. 2017]	output hex-meshes	.MESH	yes	zip
[Shang et al. 2017]	output hex-meshes	.VTK	yes	link
[Gao et al. 2017c]	input hex-meshes output hex-meshes	.VTK	no	zip
[Gao et al. 2017b]	output hex-dominant meshes	.HYBRID	no	zip
[Wu et al. 2018]	output hex-meshes	.MESH	yes	–
[Cherchi et al. 2019a]	output hex-meshes	.MESH	yes	zip
[Corman and Crane 2019]	output hex-meshes	.VTK	yes	zip
[Takayama 2019]	output hex-meshes	.VTK	yes	zip
[Gao et al. 2019]	input surface mesh input features output hex-meshes	.OBJ .FGRAPH .MESH .VTK	yes	zip
[Reberol et al. 2019]	input CAD models	.GEO	no	zip
[Yang et al. 2019]	output hex-meshes	.VTK	no	link
[Palmer et al. 2020]	input tet-mesh	.OVM	no	zip
[Livesu et al. 2020]	input surface meshes input rosy fields input features cutting loops refined surface meshes output hex-meshes	.OBJ .ROSY .SHARP .TXT .MESH	yes	github
[Guo et al. 2020]	input surface meshes input features polycubes (surface) polycubes CE (surface) output hex-meshes	.OBJ .FEA .VTK	yes	link
[Xu et al. 2021]	output hex-meshes	.MESH	yes	github
[Bukenberger et al. 2021]	output hex-meshes	.MESH	yes	–
[Pitzalis et al. 2021]	output conforming grids	.MESH	yes	–
[Ledoux 2022]	input CAD models	.STEP	no	gitlab
[Beaufort et al. 2022]	input tet-meshes input features	.VTK	no	webpage

ACKNOWLEDGMENTS

D. Bommes has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (AlgoHex, grant agreement No.853343). G. Cherchi gratefully acknowledges the support to his research by PON R&I 2014-2020 AIM1895943-1. M. Campen gratefully acknowledges funding from the Deutsche Forschungsgemeinschaft (DFG) - 427469366. A. Sheffer thanks NSERC for their ongoing support. M. Livesu was partly supported by EU ERC Advanced Grant CHANGE No. 694515.

REFERENCES

Ahmed Abdelkader, Chandrajit L Bajaj, Mohamed S Ebeida, Ahmed H Mahmoud, Scott A Mitchell, John D Owens, and Ahmad A Rushdi. 2020. VoroCrust: Voronoi meshing without clipping. *ACM Transactions on Graphics (TOG)* 39, 3 (2020), 1–16.

Table 4. List of available implementation of hex-mesh processing algorithms.

Method	Type	Sample Input Available	License	URL
[Lévy and Liu 2010]	C++	yes	–	link
[Huang et al. 2011]	executable	yes	–	zip
[Baudouin et al. 2014]	C++ (Gmsh branch)	no	GPL 2	gitlab
[Gregson et al. 2011] [Livesu et al. 2013] [Livesu et al. 2015]	executable	yes	patented, one month trial	webpage
[Fang et al. 2016]	C++	yes	–	zip
[Lyon et al. 2016]	C++	yes	GPL 3	webpage
[Fu et al. 2016]	C++ (incomplete)	no	–	zip
[Gao et al. 2017b]	C++	yes	–	github
[Gao et al. 2017a]	C++	no	–	github
[Gao et al. 2017c]	C++	no	MPL 2	github
[Xu and Chen 2018]	C++	no	GPL 3	github
[Xu et al. 2018]	C++	yes	–	github
[Liu et al. 2018]	C++	yes	GPL 3	gitlab
[Yang et al. 2019]	C++	yes	–	link
[Bracci et al. 2019]	C++ Javascript	yes	MIT	github
[Takayama 2019]	C++	no	BSD 3	bitbucket
[Gao et al. 2019]	C++	yes	–	github
[Reberol et al. 2019]	C++ (Gmsh branch)	no	GPL 2	gitlab
[Palmer et al. 2020]	Matlab	yes	MIT	github
[Verhetsel et al. 2019b]	C	no	GPL	zip
[Livesu et al. 2020]	C++	yes	GPL 3	github
[Guo et al. 2020]	C++	yes	MIT	github
[Marschner et al. 2020]	Matlab	yes	MIT	github
[Yu and Wei 2020]	C++	yes	–	github
[Pitzalis et al. 2021]	C++	no	MIT	github
[Livesu et al. 2021]	C++ (inside Cinolib)	no	MIT	github
[Neuhauser et al. 2021]	C++	yes	BSD 2	github
[Xu et al. 2021]	executable	yes	–	github
[Li et al. 2021]	C++	no	MIT	github
[Yu et al. 2022]	C++	yes	–	github

Noam Aigerman and Yaron Lipman. 2013. Injective and Bounded Distortion Mappings in 3D. *ACM Trans. Graph.* 32, 4 (2013), 1–14.

Altair. 2022. HyperMesh. https://www.altair.com/hypermesh/

ANSYS. 2022. ANSYS. https://www.ansys.com/products/meshing

Pablo Antolin, Annalisa Buffa, Elaine Cohen, John F Dannenhoffer, Gershon Elber, Stefanie Elgeti, Robert Haimes, and Richard Riesenfeld. 2019b. Optimizing microtiles in micro-structures as a design paradigm. *Computer-Aided Design* 115 (2019), 23–33.

Pablo Antolin, Annalisa Buffa, and Massimiliano Martinelli. 2019a. Isogeometric analysis on V-reps: First results. *Computer Methods in Applied Mechanics and Engineering* 355 (2019), 976–1002.

Farah Aqilah, Mazharul Islam, Franjo Juretic, Joel Guerrero, David Wood, and Farid Nasir Ani. 2018. Study of Mesh Quality Improvement for CFD Analysis of an Airfoil. *IIUM Engineering Journal* 19, 2 (2018), 203–212.

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 (2015), 70–82.

Rahul Arora, Alec Jacobson, Timothy R. Langlois, Yijiang Huang, Caitlin Mueller, Wojciech Matusik, Ariel Shamir, Karan Singh, and David I.W. Levin. 2019. Volumetric Michell Trusses for Parametric Design & Fabrication. In *Proceedings of ACM Symposium on Computation Fabrication*. 1–13.

Marco Attene. 2010. A lightweight approach to repairing digitized polygon meshes. *The visual computer* 26, 11 (2010), 1393–1406.

M. Attene, S. Biasotti, S. Bertoluzza, D. Cabiddu, M. Livesu, G. Patanè, M. Pennacchio, D. Prada, and M. Spagnuolo. 2021. Benchmarking the Geometrical Robustness of a Virtual Element Poisson Solver. *Mathematics and Computers in Simulation* 190 (2021).

Marco Attene, Marcel Campen, and Leif Kobbelt. 2013. Polygon mesh repairing: An application perspective. *Comput. Surveys* 45, 2 (2013), 1–33.

Utkarsh Ayachit. 2015. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc.

M. Balsa Rodríguez, E. Gobbetti, J.A. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S.K. Suter. 2014. State-of-the-Art in Compressed GPU-Based Direct Volume Rendering. *Computer Graphics Forum* 33, 6 (2014), 77–100.

KE Barrett. 1996. Jacobians for isoparametric finite elements. *Communications in Numerical Methods in Engineering* 12, 11 (1996), 755–766.

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 (2014), 1–30.

Arthur Bawin, François Henrotte, and Jean-François Remacle. 2021. Automatic feature-preserving size field for three-dimensional mesh generation. *Internat. J. Numer. Methods Engrg.* 122, 18 (2021), 4825–4847.

Pierre-Alexandre Beaufort, Jonathan Lambrechts, François Henrotte, Christophe Geuzaine, and Jean-François Remacle. 2017. Computing cross fields A PDE approach based on the Ginzburg-Landau theory. *Procedia engineering* 203 (2017), 219–231.

Pierre-Alexandre Beaufort, Maxence Reberol, Denis Kalmykov, Heng Liu, Franck Ledoux, and David Bommes. 2022. Hex Me If You Can. *Computer Graphics Forum* 41, 5 (2022).

L Beirão da Veiga, Franco Brezzi, Luisa Donatella Marini, and Alessandro Russo. 2014. The hitchhiker's guide to the virtual element method. *Mathematical Models and Methods in Applied Sciences* 24, 08 (2014), 1541–1573.

Steven E. Benzley, N. J. Harris, M. A, M. J. Borden Scott, and S. J. Owen. 2005a. Conformal Refinement and Coarsening of Unstructured Hexahedral Meshes. *Journal of Computing and Information Science in Engineering* 5 (2005), 330–337.

Steven E. Benzley, Nathan J. Harris, Michael Scott, Michael Borden, and Steven J. Owen. 2005b. Conformal Refinement and Coarsening of Unstructured Hexahedral Meshes. *Journal of Computing and Information Science in Engineering* 5, 4 (2005), 330–337.

Marshall W. Bern, David Eppstein, and Jeff Erickson. 2002. Flipping Cubical Meshes. *Engineering with Computers* 18, 3 (2002), 173–187.

Ted Blacker. 1996. The cooper tool. In *Proceedings of International Meshing Roundtable*.

Ted Blacker. 2000. Meeting the challenge for automated conformal hexahedral meshing. In *Proceedings of International Meshing Roundtable*. 11–20.

T.D. Blacker and R.J. Meyers. 1993. Seams and Wedges in Plastering:A 3D Hexahedral Mesh Generation Algorithm. *Engineering with Computers* 2, 9 (1993), 83–93.

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 (2013).

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

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Cláudio T. 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 (jul 2009), 10 pages.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2010. Practical mixed-integer optimization for geometry processing. In *International Conference on Curves and Surfaces*. Springer, 193–206.

Nicolas Bonneel, David Coeurjolly, Julie Digne, and Nicolas Mellado. 2020. Code replicability in computer graphics. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 93–1.

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

Michael J. Borden, Jason F. Shepherd, and Steven E. Benzley. 2002b. Mesh Cutting: Fitting Simple All-Hexahedral Meshes to Complex Geometries. In *Proceedings, 8th International Society of Grid Generation Conference*.

J. Born, P. Schmidt, and L. Kobbelt. 2021. Layout Embedding via Combinatorial Optimization. *Computer Graphics Forum* 40, 2 (2021), 277–290.

Matteo Bracci, Marco Tarini, Nico Pietroni, Marco Livesu, and Paolo Cignoni. 2019. HexaLab. net: an online viewer for hexahedral meshes. *Computer-Aided Design* 110 (2019), 24–36.

Michael L. Brewer, Lori Freitag Diachin, Patrick M. Knupp, Thomas Leurent, and Darryl J. Melander. 2003. The Mesquite Mesh Quality Improvement Toolkit. In *Proceedings of International Meshing Roundtable*.

Hendrik Brückler, David Bommes, and Marcel Campen. 2022a. Volume Parametrization Quantization for Hexahedral Meshing. *ACM Trans. Graph.* 41, 4 (2022).

Hendrik Brückler, Ojaswi Gupta, Manish Mandad, and Marcel Campen. 2022b. The 3D Motorcycle Complex for Structured Volume Decomposition. *Computer Graphics Forum* 41, 2 (2022).

Stéphanie Buchaillard, Pascal Perrier, and Yohan Payan. 2009. A biomechanical model of cardinal vowel production: Muscle activations and the impact of gravity on tongue positioning. *The Journal of the Acoustical Society of America* 126, 4 (2009),

Table 5. List of existing open source toolkits for visualization and processing of hex and hex-dominant meshes. Note that while some of these tools endow a broader set of facilities (e.g. for surface mesh processing), the table summarizes only the aspects that are relevant for the scope of this article.

Name	Type	Supported geometries	File formats	Rendering facilities	Visual inspection	Mesh attributes	Tools for volume processing	License	URL
CinoLib [Livesu 2019]	C++ library (header only)	tetrahedra, hexahedra, general polyhedra	.MESH, .VTU, .VTK, .HEDRA[1], .HEXEX[2], .HYBRID[3], Tetgen[4]	yes	plane slicing (axis aligned), thresholding, manual selection, ambient occlusion	generic attributes for all mesh elements	grid hex-meshing facilities (schemes [Livesu et al. 2021], surface mapping, faeture mapping [Gao et al. 2019]), hex-to-tet conversion [Dompierre et al. 1999], extraction of coarse block layouts, volume smoothing, subdivision schemes, padding, all quality metrics in [Stimpson et al. 2007]	MIT	github
HexaLab [Bracci et al. 2019]	web app	hexahedra	.MESH, .VTK	yes	plane slicing, thresholding, peeling, manual selection, ambient occlusion	scalar attributes (per cell)	all quality metrics in [Stimpson et al. 2007]	GPL 3	webpage github
GEOGRAM Graphite [Levy 2022a]	C++ library	tetrahedra, hexahedra, pyramids, wedges, connectors (between solid elements)	.GEOGRAM[5] .MESH	yes	plane slicing	generic attributes for all mesh elements	quality metrics, histograms	BSD 3	zip
OpenVolumeMesh [Kremer et al. 2013]	C++ library	general polyhedra	.OVM[6]	no	–	generic attributes for all mesh elements	–	GPL 3	webpage
PolyScope [Sharp et al. 2022]	C++ library	tetrahedra, hexahedra, hybrid (soup of non conforming tets and hexa)	.MESH	yes	plane slicing	scalars and vectors for all mesh elements	–	MIT	webpage
PyMesh [Zhou 2022]	Python library	tetrahedra, hexahedra	.MESH, .MSH, Tetgen[4]	yes	plane slicing	scalars and vectors for all mesh elements	–	–	webpage
Py3DViewer [Cherchi et al. 2019b]	Python library	tetrahedra, hexahedra	.MESH	yes	plane slicing	integer attributes (per cell)	–	MIT	github

[1] https://github.com/mlivesu/cinolib/blob/master/include/cinolib/io/write_HEDRA.cpp
[2] https://www.graphics.rwth-aachen.de/media/resource_files/hexex_input_examples.zip
[3] https://github.com/gaoxifeng/robust_hex_dominant_meshing/blob/master/src/meshio.cpp
[4] https://wias-berlin.de/software/tetgen/fformats.html
[5] http://alice.loria.fr/software/geogram/doc/html/geofile_8h_source.html
[6] https://www.graphics.rwth-aachen.de/media/openvolumemesh_static/Documentation/OpenVolumeMesh-Doc-Latest/file_format.html

2033–2051.
Annalisa Buffa, Ondine Chanon, and Rafael Vázquez. 2022. Analysis-aware defeaturing: Problem setting and a posteriori estimation. *Mathematical Models and Methods in Applied Sciences* 32, 02 (2022), 359–402. https://doi.org/10.1142/S0218202522500099
Dennis R Bukenberger, Marco Tarini, and Hendrik PA Lensch. 2021. At-Most-Hexa Meshes. In *Computer Graphics Forum*. Wiley Online Library.
Daniela Cabiddu, Giuseppe Patanè, and Michela Spagnuolo. 2021. PEMesh: a Graphical Framework for the Analysis of the InterplayBetween Geometry and PEM Solvers. *arXiv preprint arXiv:2102.11578* (2021).
Shengyong Cai and Timothy J Tautges. 2015. Optimizing corner assignment of submap surfaces. *Procedia Engineering* 124 (2015), 83–95.
Nestor A Calvo and Sergio R Idelsohn. 2000. All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision. *Computer Methods in Applied Mechanics and Engineering* 182, 3 (2000), 371–378.
Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Dual loops meshing: quality quad layouts on manifolds. *ACM Trans. Graph.* 31, 4 (2012), 1–11.
Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021. Efficient and Robust Discrete Conformal Equivalence with Boundary. *ACM Trans. Graph.* 40, 6 (2021).
Marcel Campen and Leif Kobbelt. 2014. Dual strip weaving: Interactive design of quad layouts using elastica strips. *ACM Trans. Graph.* 33, 6 (2014), 1–10.
Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2019. Seamless Parametrization with Arbitrary Cones for Arbitrary Genus. *ACM Trans. Graph.* 39, 1 (2019).
Marcel Campen, Claudio T. Silva, and Denis Zorin. 2016. Bijective Maps from Simplicial Foliations. *ACM Trans. Graph.* 35, 4 (2016).
Marcel Campen and Denis Zorin. 2017. Similarity Maps and Field-Guided T-Splines: A Perfect Couple. *ACM Trans. Graph.* 36, 4 (2017).
Carlos D. Carbonera and Jason F. Shepherd. 2006. A Constructive Approach to Constrained Hexahedral Mesh Generation. In *Proceedings of International Meshing Roundtable*. 435–452.
Jesse Chan, Zheng Wang, Axel Modave, Jean-Francois Remacle, and Tim Warburton. 2016. GPU-accelerated discontinuous Galerkin methods on hybrid meshes. *J. Comput. Phys.* 318 (2016), 142–168.
Alexandre Chemin, François Henrotte, Jean-François Remacle, and Jean Van Schaftingen. 2018. Representing Three-Dimensional Cross Fields Using Fourth Order Tensors. In *Proceedings of International Meshing Roundtable*, Vol. 127. 89–108.
Jinming Chen, Shuming Gao, Rui Wang, and Haiyan Wu. 2016. An approach to achieving optimized complex sheet inflation under constraints. *Computers & Graphics* 59 (2016), 39–56.
Long Chen, Gang Xu, Shiyi Wang, Zeyun Shi, and Jin Huang. 2019. Constructing volumetric parameterization based on directed graph simplification of $\ell 1$ polycube structure from complex shapes. *Computer Methods in Applied Mechanics and Engineering* 351 (2019), 422–440.
G. Cherchi, P. Alliez, R. Scateni, M. Lyon, and D. Bommes. 2019a. Selective Padding for Polycube-Based Hexahedral Meshing. *Computer Graphics Forum* 38, 1 (2019), 580–591.
Gianmarco Cherchi, Marco Livesu, and Riccardo Scateni. 2016. Polycube Simplification for Coarse Layouts of Surfaces and Volumes. *Computer Graphics Forum* 35, 5 (2016),

11–20.
Gianmarco Cherchi, Marco Livesu, Riccardo Scateni, and Marco Attene. 2020. Fast and robust mesh arrangements using floating-point arithmetic. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.
Gianmarco Cherchi, Luca Pitzalis, Giovanni Laerte Frongia, and Riccardo Scateni. 2019b. The Py3DViewer Project: A Python Library for fast Prototyping in Geometry Processing. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*.
Philippe G. Ciarlet. 2002. *Finite Element Method for Elliptic Problems*.
CoreForm. 2022a. CoreForm. https://coreform.com/products/coreform-cubit/government/
CoreForm. 2022b. Coreform Cubit Basics: Hex Meshing Fundamentals. https://www.youtube.com/watch?v=TOfq-Pknl_A
Etienne Corman and Keenan Crane. 2019. Symmetric Moving Frames. *ACM Trans. Graph.* 38, 4 (2019).
Richard Courant, Kurt Friedrichs, and Hans Lewy. 1967. On the partial difference equations of mathematical physics. *IBM journal of Research and Development* 11, 2 (1967), 215–234.
Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum (SGP)* 29, 5 (2010), 1525–1533.
Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2011. Spin Transformations of Discrete Surfaces. *ACM Trans. Graph.* 30, 4 (2011).
CUBIT. 2022. CUBIT. https://cubit.sandia.gov
Joel Daniels, Cláudio T Silva, Jason Shepherd, and Elaine Cohen. 2008. Quadrilateral mesh simplification. *ACM Trans. Graph.* 27, 5 (2008), 1–9.
David Desobry, Yoann Coudert-Osmont, Etienne Corman, Nicolas Ray, and Dmitry Sokolov. 2021. Designing 2d and 3d non-orthogonal frame fields. *Computer-Aided Design* 139 (2021), 103081.
Sailkat Dey. 1999. Curvilinear Mesh Generation in 3D. In *Proceedings of International Meshing Roundtable*. 407–417.
Lorenzo Diazzi and Marco Attene. 2021. Convex polyhedral meshing for robust solid modeling. *ACM Transactions on Graphics (TOG)* (2021).
Distene SAS. 2022. MeshGems. http://www.meshgems.com/volume-meshing-meshgems-hexa.html
Julien Dompierre, Paul Labbé, Marie-Gabrielle Vallet, and Ricardo Camarero. 1999. How to Subdivide Pyramids, Prisms, and Hexahedra into Tetrahedra. *Proceedings of International Meshing Roundtable* 99 (1999), 195.
Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. 2006. Spectral Surface Quadrangulation. *ACM Trans. Graph.* 25, 3 (2006), 1057–1066.
Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z Kovalsky, Yajie Yan, Danny M Kaufman, and Tao Ju. 2020. Lifting simplices to find injectivity. *ACM Trans. Graph.* 39, 4 (2020), 120–1.
Mohamed S Ebeida, Anjul Patney, John D Owens, and Eric Mestreau. 2011. Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates. *Internat. J. Numer. Methods Engrg.* 88, 10 (2011), 974–985.
Matthias Eck and Hugues Hoppe. 1996. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. 325–334.
Ahmed H. Elsheikh and Mustafa Elsheikh. 2014. A consistent octree hanging node elimination algorithm for hexahedral mesh generation. *Advances in Engineering Software* 75 (2014), 86 – 100.
David Eppstein. 1999. Linear complexity hexahedral mesh generation. *Computational Geometry* 12, 1-2 (1999), 3–16.
David Eppstein and Elena Mumford. 2010. Steinitz Theorems for Orthogonal Polyhedra. In *Proceedings Symposium on Computational Geometry*. 429–438.
Jeff Erickson. 2013. Theoretical advances in hexahedral mesh generation. In *Proc. 29th Annu. Symp. Comput. Geometry Workshop Mesh Gener.* 13–17.
Jeff Erickson. 2014. Efficiently Hex-Meshing Things with Topology. *Discrete & Computational Geometry* 52, 3 (2014), 427–449.
Hale Erten, Alper Üngör, and Chunchun Zhao. 2009. Mesh Smoothing Algorithms for Complex Geometric Domains. In *Proceedings of International Meshing Roundtable*. 175–193.
Vance Faber and Max Gunzburger. 1999. Centroidal Voronoi Tessellations: Applications and Algorithms. *Siam Review* 41 (1999), 637–676.
Andreas Fabri and Sylvain Pion. 2009. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 538–539.
Xianzhong Fang, Jin Huang, Yiying Tong, and Hujun Bao. 2021. Metric-Driven 3D Frame Field Generation. *IEEE Transactions on Visualization and Computer Graphics* (2021).
Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-hex Meshing Using Closed-form Induced Polycube. *ACM Trans. Graph.* 35, 4 (2016), 124:1–124:9.
N. T. Folwell and S. A. Mitchell. 1999. Reliable Whisker Weaving via Curve Contraction. *Engineering with Computers* 15, 3 (1999), 292–302.
Mihai Frâncu, Arni Asgeirsson, Kenny Erleben, and Mads JL Rønnow. 2021. Locking-Proof Tetrahedra. *ACM Transactions on Graphics (TOG)* 40, 2 (2021), 1–17.
Lori Freitag Diachin, Patrick Knupp, Todd Munson, and Suzanne Shontz. 2006. A comparison of two optimization methods for mesh quality improvement. *Engineering with Computers* 22, 2 (2006), 61–74.
Pascal Frey and Paul George. 2008. *Mesh Generation: Application to Finite Elements: Second Edition*. Iste.
Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient Volumetric PolyCube-Map Construction. *Computer Graphics Forum* 35, 7 (2016), 97–106.
Xiao-Ming Fu and Yang Liu. 2016. Computing Inversion-Free Mappings by Simplex Assembly. *ACM Trans. Graph.* 35, 6 (2016).
Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph.* 34, 4 (2015), 1–12.
Xiao-Ming Fu, Jian-Ping Su, Zheng-Yu Zhao, Qing Fang, Chunyang Ye, and Ligang Liu. 2021. Inversion-free geometric mapping construction: A survey. *Computational Visual Media* 7, 3 (2021), 289–318.
Ruiliang Gao and Jörg Peters. 2021. Improving Hexahedral-FEM-Based Plasticity in Surgery Simulation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 571–580.
Xifeng Gao and Guoning Chen. 2016. A Local Frame based Hexahedral Mesh Optimization. In *Proceedings of International Meshing Roundtable*.
Xifeng Gao, Zhigang Deng, and Guoning Chen. 2015. Hexahedral mesh re-parameterization from aligned base-complex. *ACM Trans. Graph.* 34, 4 (2015), 142.
Xifeng Gao, Jin Huang, Kaoji Xu, Zherong Pan, Zhigang Deng, and Guoning Chen. 2017a. Evaluating Hex-mesh Quality Metrics via Correlation Analysis. *Computer Graphics Forum* 36, 5 (2017), 105–116.
Xifeng Gao, Wenzel Jakob, Marco Tarini, and Daniele Panozzo. 2017b. Robust Hex-dominant Mesh Generation Using Field-guided Polyhedral Agglomeration. *ACM Trans. Graph.* 36, 4 (2017), 114:1–114:13.
Xifeng Gao, Tobias Martin, Sai Deng, Elaine Cohen, Zhigang Deng, and Guoning Chen. 2016. Structured Volume Decomposition via Generalized Sweeping. *IEEE Transactions on Visualization and Computer Graphics* 22, 7 (2016), 1899–1911.
Xifeng Gao, Daniele Panozzo, Wenping Wang, Zhigang Deng, and Guoning Chen. 2017c. Robust structure simplification for hex re-meshing. *ACM Trans. Graph.* 36, 6 (2017), 185.
Xifeng Gao, Hanxiao Shen, and Daniele Panozzo. 2019. Feature Preserving Octree-Based Hexahedral Meshing. *Computer Graphics Forum* 38, 5 (2019), 135–149.
Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. 2021. Foldover-Free Maps in 50 Lines of Code. *ACM Trans. Graph.* 40, 4 (2021).
Jean-Michel Gérard, Reiner Wilhelms-Tricarico, Pascal Perrier, and Yohan Payan. 2006. A 3D dynamical biomechanical tongue model to study speech motor control. *arXiv preprint physics/0606148* (2006).
Christophe Geuzaine and Jean-François Remacle. 2009. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *Internat. J. Numer. Methods Engrg.* 79, 11 (2009), 1309–1331.
Mark Gillespie, Boris Springborn, and Keenan Crane. 2021. Discrete Conformal Equivalence of Polyhedral Surfaces. *ACM Trans. Graph.* 40, 4 (2021).
Nadja Goerigk and Hang Si. 2015. On Indecomposable Polyhedra and the Number of Steiner Points. *Procedia Engineering* 124 (2015), 343–355.
Dmitry Golovaty, Jose Alberto Montero, and Daniel Spirn. 2021. A Variational Method for Generating n-Cross Fields Using Higher-Order Q-Tensors. *SIAM Journal on Scientific Computing* 43, 5 (2021), A3269–A3304. https://doi.org/10.1137/19M1287857
Jens Gravesen, Anton Evgrafov, Dang-Manh Nguyen, and Peter Nørtoft. 2014. Planar Parametrization in Isogeometric Analysis. In *Mathematical Methods for Curves and Surfaces*. 189–212.
Anthony Gravouil, Thomas Elguedj, and Hubert Maigre. 2009. An explicit dynamics extended finite element method. Part 2: Element-by-element stable-explicit/explicit dynamic scheme. *Computer Methods in Applied Mechanics and Engineering* 198, 30-32 (2009), 2318–2328.
James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Computer Graphics Forum* 30, 5 (2011), 1407–1416.
GRSI. 2022. Graphics Replicability Stamp Initiative. http://www.replicabilitystamp.org
Hao-Xiang Guo, Xiaohan Liu, Dong-Ming Yan, and Yang Liu. 2020. Cut-enhanced PolyCube-maps for feature-aware all-hex meshing. *ACM Trans. Graph.* 39, 4 (2020), 106–1.
Allen Hatcher. 2000. *Algebraic topology*. Cambridge Univ. Press, Cambridge.
Alejo Hausner. 2001. Simulating Decorative Mosaics. In *Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. 573–580.
Victoria Hernandez-Mederos, Jorge Estrada-Sarlabous, and Dionne Leon Madrigal. 2006. On local injectivity of 2D triangular cubic Bezier functions. *Investigación Operacional* 27, 3 (2006), 261–276.
Kai Hormann, Konrad Polthier, and Alla Sheffer. 2008. Mesh Parameterization: Theory and Practice. In *ACM SIGGRAPH ASIA 2008 Courses*.
Kangkang Hu, Jin Qian, and Yongjie Jessica Zhang. 2013. Adaptive All-Hexahedral Mesh Generation Based on A Hybrid Octree and Bubble Packing. In *Proceedings of*

International Meshing Roundtable.

Kangkang Hu and Yongjie Jessica Zhang. 2016. Centroidal Voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering* 305 (2016), 405 – 421.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast tetrahedral meshing in the wild. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 117–1.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60–1.

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 (2014), 25:1–25:11.

Jin Huang, Yiying Tong, Hongyu Wei, and Hujun Bao. 2011. Boundary Aligned Smooth 3D Cross-Frame Field. *ACM Trans. Graph.* 30, 6 (2011), 1–8.

Thomas JR Hughes, John A Cottrell, and Yuri Bazilevs. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 39-41 (2005), 4135–4195.

Yasushi Ito, Alan M Shih, and Bharat K Soni. 2009. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *Internat. J. Numer. Methods Engrg.* 77, 13 (2009), 1809–1833.

Steven R. Jankovich, Steven E. Benzley, Jason F. Shepherd, and Scott A. Mitchell. October 1999. The Graft Tool: An All-Hexahedral Transition Algorithm for Creating Multi-directional Swept Volume Mesh. In *Proceedings of International Meshing Roundtable*. 387–392.

Tengfei Jiang, Jin Huang, Yuanzhen Wang, Yiying Tong, and Hujun Bao. 2014. Frame field singularity correction for automatic hexahedralization. *IEEE Transactions on Visualization and Computer Graphics* 20, 8 (2014), 1189–1199.

Amaury Johnen, J-F Remacle, and Christophe Geuzaine. 2013. Geometrical validity of curvilinear finite elements. *J. Comput. Phys.* 233 (2013), 359–372.

Amaury Johnen, J-C Weill, and J-F Remacle. 2017. Robust and efficient validation of the linear hexahedral element. *Procedia Engineering* 203 (2017), 271–283.

Katerina Jurkova, Franck Ledoux, Raphael Kuate, Thomas Rickmeyer, Timothy J. Tautges, and Hamdi Zorgati. 2008. Local Topological Modifications of Hexahedral Meshes; Part II: Combinatorics and Relation To Boy Surface. In *ESAIM Proceedings*, Vol. 24. 34–45.

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

Yasumi Kawamura, Md. Shahidul Islam, and Yoichi Sumi. 2008. A strategy of automatic hexahedral mesh generation by using an improved whisker-weaving method with a surface mesh modification procedure. *Engineering with Computers* 24 (2008), 215–219.

Alan Kelly, Lukasz Kaczmarczyk, and Chris Pearce. 2013. Mesh Improvement Methodology for 3D Volumes with Non-planar Surfaces. In *Proceedings of International Meshing Roundtable*. 55–69.

Ho-Young Kim and Hyun-Gyu Kim. 2021. A hexahedral-dominant FE meshing technique using trimmed hexahedral elements preserving sharp edges and corners. *Engineering with Computers* (10 2021). https://doi.org/10.1007/s00366-021-01526-0

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 1–10.

P. Knupp. 1998. Next-generation sweep tool: A method for generating all-hex meshes on two-and-a-half dimensional geometries. In *Proceedings of International Meshing Roundtable*. 505–513.

Patrick Knupp. 2007. *Remarks on Mesh Quality*. Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Patrick Knupp, Tzanio Kolev, Ketan Mittal, and Vladimir Z. Tomov. 2021. Adaptive Surface Fitting and Tangential Relaxation for High-Order Mesh Optimization. (2021). arXiv:2105.12165

Patrick M Knupp. 1990. On the invertibility of the isoparametric map. *Computer Methods in Applied Mechanics and Engineering* 78, 3 (1990), 313–329.

Patrick M. Knupp. 2000. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II—A framework for volume mesh optimization and the condition number of the Jacobian matrix. *Internat. J. Numer. Methods Engrg.* 48, 8 (2000), 1165–1185.

Patrick M. Knupp. 2001a. Algebraic Mesh Quality Metrics. *SIAM Journal on Scientific Computing* 23, 1 (2001), 193–218.

P. M. Knupp. 2001b. Hexahedral and Tetrahedral Mesh Untangling. *Engineering with Computers* 17, 3 (2001), 261–268.

Patrick M. Knupp. 2003. A method for hexahedral mesh shape optimization. *Internat. J. Numer. Methods Engrg.* 58, 2 (2003), 319–332.

Julien Dompierre Ko-Foa Tchon and Ricardo Camarero. 2002. Conformal Refinement of All-Quadrilateral and All-Hexahedral Meshes According to an Anisotropic Metric. In *Proceedings of International Meshing Roundtable*. 231–242.

Nicolas Kowalski, Franck Ledoux, and Pascal Frey. 2016. Smoothness driven frame field generation for hexahedral meshing. *Computer-Aided Design* 72 (2016), 65–77.

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

Michael Kremer, David Bommes, and Leif Kobbelt. 2013. OpenVolumeMesh–A versatile index-based data structure for 3D polytopal complexes. In *Proceedings of the 21st International Meshing Roundtable*. Springer, 531–548.

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

Mingwu Lai, Steven Benzley, and David White. 2000. Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. *Internat. J. Numer. Methods Engrg.* 49, 1-2 (2000), 261–275.

Frank Ledoux. 2022. MAMBO. https://gitlab.com/franck.ledoux/mambo

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

Franck Ledoux and Jean-Christophe Weill. 2008. An Extension of the Reliable Whisker Weaving Algorithm. In *Proceedings of International Meshing Roundtable*. 215–232.

Na Lei, Xiaopeng Zheng, Jian Jiang, Yu-Yao Lin, and David Xianfeng Gu. 2017. Quadrilateral and hexahedral mesh generation based on surface foliation theory. *Computer Methods in Applied Mechanics and Engineering* 316 (2017), 758–781. Special Issue on Isogeometric Analysis: Progress and Challenges.

Richard Leroy. 2008. *Certificates of positivity and polynomial minimization in the multivariate Bernstein basis*. Ph.D. Dissertation. University of Rennes 1.

Bruno Levy. 2022a. GeoGram. http://alice.loria.fr/software/geogram/

Bruno Levy. 2022b. Graphite. http://alice.loria.fr/software/graphite/doc/html/.

Bruno Lévy. 2022. Partial optimal transport for a constant-volume Lagrangian mesh with free boundaries. *J. Comput. Phys.* 451 (2022), 110838.

Bruno Lévy and Yang Liu. 2010. L p Centroidal Voronoi Tessellation and its applications. *ACM Trans. Graph.* 29, 4 (2010), 1–11.

Bo Li, Xin Li, Kexiang Wang, and Hong Qin. 2010. Generalized PolyCube Trivariate Splines. In *Shape Modeling International*. 261–265.

Bo Li, Xin Li, Kexiang Wang, and Hong Qin. 2013. Surface mesh to volumetric spline conversion with generalized polycubes. *IEEE Transactions on Visualization and Computer Graphics* 19, 9 (2013), 1539–1551.

Lingxiao Li, Paul Zhang, Dmitriy Smirnov, S Mazdak Abulnaga, and Justin Solomon. 2021. Interactive All-Hex Meshing via Cuboid Decomposition. *ACM Trans. Graph.* 40, 6 (2021).

TS Li, RM McKeag, and CG Armstrong. 1995. Hexahedral meshing using midpoint subdivision and integer programming. *Computer Methods in Applied Mechanics and Engineering* 124, 1-2 (1995), 171–193.

Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. 2012. All-Hex Meshing Using Singularity-Restricted Field. *ACM Trans. Graph.* (2012).

Hongwei Lin, Sinan Jin, Hongwei Liao, and Qun Jian. 2015. Quality Guaranteed All-hex Mesh Generation by a Constrained Volume Iterative Fitting Algorithm. *Computer-Aided Design* 67, C (2015), 107–117.

Konstantin Lipnikov. 2013. On shape-regularity of polyhedral meshes for solving PDEs. In *Proceedings of International Meshing Roundtable*.

Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. 2018. Singularity-constrained octahedral fields for hexahedral meshing. *ACM Trans. Graph.* 37, 4 (2018), 93.

Jianfei Liu, Shuli Sun, and Yongqiang Chen. 2007. SPR – A New Method for Mesh Improvement and Boundary Recovery. In *Computational Mechanics*. Springer, 180–186.

Lei Liu, Yongjie Zhang, Yang Liu, and Wenping Wang. 2015. Feature-preserving T-mesh construction using skeleton-based polycubes. *Computer-Aided Design* 58 (2015), 162–172.

Shang-sheng Liu and Rajit Gadh. 1997. Automatic Hexahedral Mesh Generation by Recursive Convex and Swept Volume Decomposition. In *Proceedings of International Meshing Roundtable*. 217–231.

S.-S. Liu, J. Uicker, and R. Gadh. 1999. A dual geometry—topology constraint approach for determination of pseudo-swept shapes as applied to hexahedral mesh generation. *Computer-Aided Design* 31, 6 (1999), 413 – 426.

Marco Livesu. 2019. Cinolib: A Generic Programming Header Only C++ Library for Processing Polygonal and Polyhedral Meshes. In *Transactions on Computational Science XXXIV*. 64–76.

Marco Livesu, Marco Attene, Giuseppe Patane, and Michela Spagnuolo. 2017. Explicit Cylindrical Maps for General Tubular Shapes. *Computer-Aided Design* 90 (2017), 27 – 36.

Marco Livesu, Alessandro Muntoni, Enrico Puppo, and Riccardo Scateni. 2016. Skeleton-driven Adaptive Hexahedral Meshing of Tubular Shapes. *Computer Graphics Forum* 35, 7 (2016), 237–246.

Marco Livesu, Nico Pietroni, Enrico Puppo, Alla Sheffer, and Paolo Cignoni. 2020. LoopyCuts: Practical Feature-Preserving Block Decomposition for Strongly Hex-Dominant Meshing. *ACM Trans. Graph.* 39, 4 (2020).

Marco Livesu, Luca Pitzalis, and Gianmarco Cherchi. 2021. Optimal Dual Schemes for Adaptive Grid Based Hexmeshing. *ACM Trans. Graph.* (2021).

Marco Livesu, Alla Sheffer, Nicholas Vining, and Marco Tarini. 2015. Practical Hex-mesh Optimization via Edge-cone Rectification. *ACM Trans. Graph.* 34, 4 (2015), 141:1–141:11.

Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. 2013. PolyCut: Monotone Graph-cuts for PolyCube Base-complex Construction. *ACM Trans. Graph.* 32, 6 (2013), 171:1–171:12.

Claudio Lobos, Cristopher Arenas, Esteban Daines, and Nancy Hitschfeld. 2021. Measuring geometrical quality of different 3D linear element types. *Numerical Algorithms* (2021), 1–24.

Jean Hsiang-Chun Lu, William Roshan Quadros, and Kenji Shimada. 2017. Evaluation of user-guided semi-automatic decomposition tool for hexahedral mesh generation. *Journal of Computational Design and Engineering* 4, 4 (2017), 330–338.

Yong Lu, Rajit Gadh, and Timothy J Tautges. 2001. Feature based hex meshing methodology: feature recognition and volume decomposition. *Computer-Aided Design* 33, 3 (2001), 221–232.

Xiaojuan Luo, Mark S. Shephard, Jean-François Remacle, Robert M. O'Bara, Mark W. Beall, Barna A. Szabó, and Ricardo Actis. 2002. p-Version Mesh Generation Issues. In *Proceedings of International Meshing Roundtable*. 343–354.

Max Lyon, David Bommes, and Leif Kobbelt. 2016. HexEx: robust hexahedral mesh extraction. *ACM Trans. Graph.* 35, 4 (2016), 123.

John B. Malone. 2012. *Two-Refinement by Pillowing for Structured Hexahedral Meshes*. Ph.D. Dissertation. BYU.

Manish Mandad and Marcel Campen. 2020. Efficient piecewise higher-order parametrization of discrete surfaces with local and global injectivity. *Computer-Aided Design* 127 (2020), 102862.

Manish Mandad, Ruizhi Chen, David Bommes, and Marcel Campen. 2022. Intrinsic mixed-integer polycubes for hexahedral meshing. *Computer Aided Geometric Design* 94 (2022).

Loïc Maréchal. 2009. Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In *Proceedings of International Meshing Roundtable*. 65–84.

Z. Marschner, D. Palmer, P. Zhang, and J. Solomon. 2020. Hexahedral Mesh Repair via Sum-of-Squares Relaxation. *Computer Graphics Forum* 39, 5 (2020), 133–147.

Fady Massarwi and Gershon Elber. 2016. A B-spline based framework for volumetric object modeling. *Computer-Aided Design* 78 (2016), 36–47.

Karl Merkley, Corey D. Ernst, Jason F. Shepherd, and M. J. Borden. 2007. Methods and Applications of Generalized Sheet Insertion for Hexahedral Meshing. In *Proceedings of International Meshing Roundtable*. 233–250.

Sia Meshkat and Dafna Talmor. 2000. Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. *Internat. J. Numer. Methods Engrg.* 49, 1-2 (2000), 17–30.

Ray J. Meyers and Timothy J. Tautges. 1998. The "Hex-Tet" Hex-Dominant Meshing Algorithm as Implemented in CUBIT. In *IMR*.

AR Mitchell, G Phillips, and E Wachspress. 1971. Forbidden shapes in the finite element method. *IMA Journal of Applied Mathematics* 8, 2 (1971), 260–269.

Scott A. Mitchell. 1996. A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volume. In *Proceedings of STACS 96*. 465–476.

Scott A. Mitchell and Timothy J. Tautges. 1995. Pillowing Doublets: Refining a Mesh to Ensure That Faces Share At Most One Edge. In *Proceedings of International Meshing Roundtable*.

Scott A Mitchell and Stephen A Vavasis. 1992. Quality mesh generation in three dimensions. In *Proc. eighth annual symposium on Computational Geometry*. 212–221.

Katsuhiro Miyoshi and Ted D. Blacker. 2000. Hexahedral Mesh Generation Using Multi-Axis Cooper Algorithm. In *Proceedings of International Meshing Roundtable*. 89–97.

ADOS Moraes, P Lage, G Cunha, and LFLR da Silva. 2013. Analysis of the non-orthogonality correction of finite volume discretization on unstructured meshes. In *Proceedings of the 22nd International Congress of Mechanical Engineering*. 3,7.

Lin Mu, Xiaoshen Wang, and Yanqiu Wang. 2015. Shape regularity conditions for polygonal/polyhedral meshes, exemplified in a discontinuous Galerkin discretization. *Numerical Methods for Partial Differential Equations* 31, 1 (2015), 308–325.

Matthias Müller-Hannemann. 2001. Shelling hexahedral complexes for mesh generation. *Graph Algorithms and Applications* 5, 5 (2001), 59–91.

M. Müller-Hannemann. 2002. Quadrilateral surface meshes without self-intersecting dual cycles for hexahedral mesh generation. *Computational Geometry* 22 (2002), 75–97.

Peter Murdoch, Steven Benzley, Ted Blacker, and Scott A. Mitchell. 1997. The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes. *Finite Elements in Analysis and Design* 28, 2 (1997), 137 – 149.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-Aligned Global Parametrization. *ACM Trans. Graph.* 33, 4 (2014).

Christoph Alexander Neuhauser, Junpeng Wang, and Ruediger Westermann. 2021. Interactive Focus+Context Rendering for Hexahedral Mesh Inspection. *IEEE Transactions on Visualization and Computer Graphics* (2021), 1–1.

M. Nieser, U. Reitebuch, and K. Polthier. 2011. CubeCover–Parameterization of 3D Volumes. *Computer Graphics Forum* 30, 5 (2011), 1397–1406.

Stefano Nuvoli, Alex Hernandez, Claudio Esperança, Riccardo Scateni, Paolo Cignoni, and Nico Pietroni. 2019. QuadMixer: Layout Preserving Blending of Quadrilateral Meshes. *ACM Trans. on Graphics* 38, 6 (2019).

Steven Owen and Saigal Sunil. 2000. H-Morph: An Indirect Approach to Advancing Front Hex Meshing. *Internat. J. Numer. Methods Engrg.* 1, 49 (2000), 289–312.

Steven J Owen. 1998. A survey of unstructured mesh generation technology. *Proceedings of International Meshing Roundtable*, 239–267.

Steven J Owen, Ryan M Shih, and Corey D Ernst. 2017. A template-based approach for parallel hexahedral two-refinement. *Computer-Aided Design* 85 (2017), 34–52.

Gilles-Philippe Paillé, Pierre Poulin, and Bruno Lévy. 2013. Fitting Polynomial Volumes to Surface Meshes with Voronoï Squared Distance Minimization. *Computer Graphics Forum* 32, 5 (2013), 103–112.

Gilles-Philippe Paillé, Nicolas Ray, Pierre Poulin, Alla Sheffer, and Bruno Lévy. 2015. Dihedral Angle-Based Maps of Tetrahedral Meshes. *ACM Trans. Graph.* 34, 4 (2015).

David Palmer, David Bommes, and Justin Solomon. 2020. Algebraic Representations for Volumetric Frame Fields. *ACM Trans. Graph.* 39, 2 (2020).

Michael Parrish, M. J. Borden, M. L. Staten, and S. E. Benzley. 2007. A Selective Approach to Conformal Refinement of Unstructured Hexahedral Finite Element Meshes. In *Proceedings of International Meshing Roundtable*. 251–268.

Jeanne Pellerin, Amaury Johnen, and Jean-Francois Remacle. 2017. Identifying combinations of tetrahedra into hexahedra: a vertex based strategy. *Procedia Engineering* 203 (2017), 2–13.

Chi-Han Peng, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. 2011. Connectivity Editing for Quadrilateral Meshes. *ACM Trans. Graph.* 30, 6 (2011), 1–12.

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable feature-line driven quad-remeshing. *ACM Trans. on Graphics* 40, 4 (2021).

Luca Pitzalis, Marco Livesu, Gianmarco Cherchi, Enrico Gobbetti, and Riccardo Scateni. 2021. Generalized Adaptive Refinement for Grid-based Hexahedral Meshing. *ACM Transactions on Graphics (SIGGRAPH Asia)* 40, 6 (2021). https://doi.org/10.1145/3478513.3480508

Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. 2002. *Bezier and B-Spline Techniques*. Springer-Verlag.

Hartmut Prautzsch and Leif Kobbelt. 1994. Convergence of subdivision and degree elevation. *Advances in Computational Mathematics* 2, 1 (1994), 143–154.

M.A. Price and C.G. Armstrong. 1997. Hexahedral mesh generation by medial surface subdivision: Part II. solids with flat and concave edges. *Internat. J. Numer. Methods Engrg.* 40, 1 (1997), 111–136.

M.A. Price, C.G. Armstrong, and M.A. Sabin. 1995. Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges. *Internat. J. Numer. Methods Engrg.* 38, 19 (1995), 3335–3359.

François Protais, Maxence Reberol, Nicolas Ray, Etienne Corman, Franck Ledoux, and Dmitry Sokolov. 2022. Robust Quantization for Polycube Maps. *Computer-Aided Design* 150 (2022).

Jin Qian and Yongjie Zhang. 2010. Sharp Feature Preservation in Octree-Based Hexahedral Mesh Generation for CAD Assembly Models. In *Proceedings of International Meshing Roundtable*. 243–262.

Jin Qian and Yongjie Zhang. 2012. Automatic unstructured all-hexahedral mesh generation from B-Reps for non-manifold CAD assemblies. *Engineering with Computers* 28, 4 (2012), 345–359.

William Roshan Quadros. 2014. LayTracks3D: A New Approach to Meshing General Solids using Medial Axis Transform. *Procedia Engineering* 82 (2014), 72 – 87.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable locally injective mappings. *ACM Trans. Graph.* 36, 4 (2017), 1.

Nicolas Ray, Dmitry Sokolov, and Bruno Lévy. 2016. Practical 3D frame field generation. *ACM Trans. Graph.* 35, 6 (2016), 233.

Nicolas Ray, Dmitry Sokolov, Maxence Reberol, Franck Ledoux, and Bruno Levy. 2018. Hex-dominant meshing: mind the gap! *Computer-Aided Design* 102 (2018), 94–103.

Faniry H Razafindrazaka and Konrad Polthier. 2017. Optimal base complexes for quadrilateral meshes. *Computer Aided Geometric Design* 52 (2017), 63–74.

Maxence Reberol, Alexandre Chemin, and Jean-François Remacle. 2019. Multiple Approaches to Frame Field Correction for CAD Models. *arXiv preprint arXiv:1912.01248* (2019).

Maxence Reberol, Kilian Verhetsel, François Henrotte, David Bommes, and Jean-François Remacle. 2021. Robust topological construction of all-hexahedral boundary layer meshes. (2021).

Xevi Roca, Abel Gargallo-Peiró, and Josep Sarrate. 2012. Defining Quality Measures for High-Order Planar Triangles and Curved Mesh Generation. In *Proceedings of International Meshing Roundtable*. 365–383.

X. Roca and J. Sarrate. 2008. Local Dual Contributions on Simplices: A Tool for Block Meshing. In *Proceedings of International Meshing Roundtable*. 513–531.

P-Y Rohan, Claudio Lobos, Mohammad Ali Nazari, Pascal Perrier, and Yohan Payan. 2017. Finite element models of the human tongue: a mixed-element mesh approach.

Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 5, 6 (2017), 390–400.

Eloi Ruiz-Gironés, Xevi Roca, and Josep Sarrate. 2011. Using a computational domain and a three-stage node location procedure for multi-sweeping algorithms. *Advances in Engineering Software* 42, 9 (2011), 700–713.

Eloi Ruiz-Gironés, Xevi Roca, and Josep Sarrate. 2012. The receding front method applied to hexahedral mesh generation of exterior domains. *Engineering with Computers* 28, 4 (2012), 391–408.

Eloi Ruiz-Gironés, Xevi Roca, and Jose Sarrate. 2014. Optimizing Mesh Distortion by Hierarchical Iteration Relocation of the Nodes on the CAD Entities. *Procedia Engineering* 82 (2014), 101–113.

E. Ruiz-Gironés, X. Roca, J. Sarrate, R. Montenegro, and J.M. Escobar. 2015. Simultaneous untangling and smoothing of quadrilateral and hexahedral meshes using an object-oriented framework. *Advances in Engineering Software* 80 (2015), 12–24.

Malcolm Arthur Sabin. 1997. *Spline finite elements.* Ph.D. Dissertation. Leeds University.

Raviprakash R Salagame and Ashok D Belegundu. 1994. Distortion, degeneracy and rezoning in finite elements—A survey. *Sadhana* 19, 2 (1994), 311–335.

Josep Sarrate Ramos, Eloi Ruiz-Gironés, and Francisco Javier Roca Navarro. 2014. Unstructured and semi-structured hexahedral mesh generation methods. *Computational Technology Reviews* 10 (2014), 35–64.

Shankar Sastry and Suzanne Shontz. 2014. A parallel log-barrier method for mesh quality improvement and untangling. *Engineering with Computers* 30, 4 (2014), 503–515.

Shankar Prasad Sastry and Suzanne M. Shontz. 2009. A Comparison of Gradient- and Hessian-Based Optimization Methods for Tetrahedral Mesh Quality Improvement. In *Proceedings of International Meshing Roundtable*. 631–648.

M. Scherer, R. Denzer, and P. Steinmann. 2010. A fictitious energy approach for shape optimization. *Internat. J. Numer. Methods Engrg.* 82, 3 (2010), 269–302.

Teseo Schneider, Jeremie Dumas, Xifeng Gao, Mario Botsch, Daniele Panozzo, and Denis Zorin. 2019. Poly-Spline Finite-Element Method. *ACM Trans. Graph* 38, 3 (2019), 19:1–19:16.

Teseo Schneider, Yixin Hu, Jérémie Dumas, Xifeng Gao, Daniele Panozzo, and Denis Zorin. 2018. Decoupling simulation accuracy from mesh quality. *ACM transactions on graphics* (2018).

Teseo Schneider, Yixin Hu, Xifeng Gao, Jérémie Dumas, Denis Zorin, and Daniele Panozzo. 2022. A Large-Scale Comparison of Tetrahedral and Hexahedral Elements for Solving Elliptic PDEs with the Finite Element Method. *ACM Trans. Graph.* 41, 3, Article 23 (mar 2022), 14 pages. https://doi.org/10.1145/3508372

R. Schneiders. 1996a. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers* 12, 3 (1996), 168–177.

Robert Schneiders. 1996b. Refining quadrilateral and hexahedral element meshes. *transition* 2 (1996), 1.

R. Schneiders. 1997. An algorithm for the generation of hexahedral element meshes based on a octree technique. In *Proceedings of International Meshing Roundtable*. 183–194.

R. Schneiders. 1999. Quadrilateral and Hexahedral Element Meshes. In *Handbook of Grid Generation*, Nigel P. Weatherill Joe F. Thompson, Bharat K. Soni (Ed.). CRC Press, Chapter 21.

Robert Schneiders. 2000. Algorithms for quadrilateral and hexahedral mesh generation. *Proceedings of the VKI Lecture Series on Computational Fluid Dynamic* 4 (2000).

Robert Schneiders and Rolf Bünten. 1995. Automatic generation of hexahedral finite element meshes. *Computer Aided Geometric Design* 12, 7 (1995), 693 – 707.

R. Schneiders, R. Schindler, and F. Weiler. 1996. Octree-based Generation of Hexahedral Element Meshes. In *Proceedings of International Meshing Roundtable*. 205–215.

Alexandra Schonning, Binu Oommen, Irina Ionescu, and Ted Conway. 2009. Hexahedral mesh development of free-formed geometry: The human femur exemplified. *Computer-Aided Design* 41, 8 (2009), 566–572.

Will Schroeder, Kenneth M Martin, and William E Lorensen. 1998. *The visualization toolkit an object-oriented approach to 3D graphics.* Prentice-Hall, Inc.

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013), 125–135.

M.A. Scott, S.E. Benzley, and S.J. Owen. 2006. Improved many-to-one sweeping. *Internat. J. Numer. Methods Engrg.* 65 (2006), 332–348.

Feifei Shang, Yangke Gan, and Yufei Guo. 2017. Hexahedral mesh generation via constrained quadrilateralization. *PLOS ONE* 12, 5 (2017), 1–27.

Nicholas Sharp et al. 2022. Polyscope. www.polyscope.run.

Alla Sheffer, Michal Etzion, Ari Rappoport, and Michel Bercovier. 1999. Hexahedral mesh generation using the embedded Voronoi graph. *Engineering with Computers* 15, 3 (1999), 248–262.

Chun Shen, Shuming Gao, and Rui Wang. 2021. Topological operations for editing the singularity on a hex mesh. *Engineering with Computers* 37, 2 (2021), 1357–1375.

Chun Shen, Shuming Gao, and Rui Wang. 2022a. Hexahedral mesh adaptation based on posterior-error estimation. *Engineering with Computers* (2022), 1–12.

Hanxiao Shen, Leyi Zhu, Ryan Capouellez, Daniele Panozzo, Marcel Campen, and Denis Zorin. 2022b. Which Cross Fields can be Quadrangulated? Global Parameterization from Prescribed Holonomy Signatures. *ACM Trans. Graph.* 41, 4 (2022).

Jason F. Shepherd. 2007. *Topologic and Geometric Constraint-Based Hexahedral Mesh Generation.* Ph.D. Dissertation. School of Computing. Advisor(s) Johnson, Chris R.

Jason F Shepherd, Mark W Dewey, Adam C Woodbury, Steven E Benzley, Matthew L Staten, and Steven J Owen. 2010. Adaptive mesh coarsening for quadrilateral and hexahedral meshes. *Finite Elements in Analysis and Design* 46, 1-2 (2010), 17–32.

J. F. Shepherd and C. R. Johnson. 2008. Hexahedral mesh generation constraints. *Engineering with Computers* 24, 3 (2008), 195–213.

Jonathan Shewchuk. 2002. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). (2002).

B. Shih and H. Sakurai. 1996. Automated Hexahedral Mesh Generation by Swept Volume Decomposition and Recomposition. In *Proceedings of International Meshing Roundtable*.

K. Shimada and S. Yamakawa. 2002. Hex-Dominant Mesh Generation with Directionality Control via Packing Rectangular Solid Cells. In *Geometric Modeling and Processing*. 107.

Dmitry Sokolov. 2016. *Modélisation géométrique.* habilthesis. Université de Lorrainee.

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

Dmitry Sokolov, Nicolas Ray, Lionel Untereiner, and Bruno Lévy. 2016. Hexahedral-Dominant Meshing. *ACM Trans. Graph.* 35, 5 (2016), 157:1–157:23.

Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018. Optimal cone singularities for conformal flattening. *ACM Trans. Graph.* 37, 4 (2018), 1–17.

Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary element octahedral fields in volumes. *ACM Trans. Graph.* 36, 3 (2017), 28.

Tommaso Sorgente, Silvia Biasotti, Gianmarco Manzini, and Michela Spagnuolo. 2022. The role of mesh quality and mesh quality indicators in the virtual element method. *Advances in Computational Mathematics* 48, 1 (2022), 1–34.

Tommaso Sorgente, Daniele Prada, Daniela Cabiddu, Silvia Biasotti, Giuseppe Patane, Micol Pennacchio, Silvia Bertoluzza, Gianmarco Manzini, and Michela Spagnuolo. 2021. VEM and the Mesh. *arXiv preprint arXiv:2103.01614* (2021).

Matthew Staten, Scott Canann, and Steven Owen. 1999. BMSweep: Locating Interior Nodes During Sweeping. *Engineering with Computers* 15 (1999), 212–218.

M. Staten and K. Shimada. 2010. A close look at valences in hexahedral element meshes. *Internat. J. Numer. Methods Engrg.* 83 (2010), 899–914.

Matthew L. Staten, Robert A. Kerr, Steven J. Owen, and Ted D. Blacker. 2006. Unconstrained Paving and Plastering: Progress Update. In *Proceedings of International Meshing Roundtable*.

Matthew L. Staten, Robert A. Kerr, Steven J. Owen, Ted D. Blacker, Marco Stupazzini, and Kenji Shimada. 2010a. Unconstrained plastering - Hexahedral mesh generation via advancing-front geometry decomposition. *Internat. J. Numer. Methods Engrg.* 81 (2010), 135–171.

Matthew L. Staten, Steven J. Owen, and Ted D. Blacker. 2005. Unconstrained Paving & Plastering: A New Idea for All Hexahedral Mesh Generation. In *Proceedings of International Meshing Roundtable*. 399–416.

M. L. Staten, J. F. Shepherd, F. Ledoux., and K. Shimada. 2010b. Hexahedral Mesh Matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces. *Internat. J. Numer. Methods Engrg.* 82, 12 (2010), 1475–1509.

CJ Stimpson, CD Ernst, P Knupp, PP Pébay, and D Thompson. 2007. The Verdict library reference manual. *Sandia National Laboratories Technical Report* 9 (2007).

Florian Cyril Stutz, Tim Felle Olsen, Jeroen Peter Groen, Tuan Nguyen Trung, Niels Aage, Ole Sigmund, Justin Solomon, and Jakob Andreas Bærentzen. 2022. Synthesis of Frame Field-Aligned Multi-Laminar Structures. *ACM Trans. Graph.* 41, 5, Article 170 (may 2022), 20 pages. https://doi.org/10.1145/3516522

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

Lu Sun, Guoqun Zhao, and Xinwu Ma. 2012. Quality improvement methods for hexahedral element meshes adaptively generated using grid-based algorithm. *Internat. J. Numer. Methods Engrg.* 89, 6 (2012), 726–761.

LH Tack, R Schneiders, J Debye, R Kopp, and W Oberschelp. 1994. Two-and three-dimensional remeshing, mesh refinement and application to simulation of micromechanical processes. *Computational Materials Science* 3, 2 (1994), 241–246.

Kenshi Takayama. 2019. Dual Sheet Meshing: An Interactive Approach to Robust Hexahedralization. *Computer Graphics Forum* 38, 2 (2019), 37–48.

Erik G Takhounts, Stephen A Ridella, Vikas Hasija, Rabih E Tannous, J Quinn Campbell, Dan Malone, Kerry Danelson, Joel Stitzel, Steve Rowson, and Stefan Duma. 2008. Investigation of traumatic brain injuries using the next generation of simulated injury monitor (SIMon) finite element head model. *Stapp car crash journal* 52 (2008), 1.

Michael Tao, Christopher Batty, Eugene Fiume, and David IW Levin. 2019. Mandoline: robust cut-cell generation for arbitrary triangle meshes. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–17.

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

Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. 2011. Simple Quad Domains for Field Aligned Mesh Parametrization. *ACM Trans. Graph.*

30, 6 (2011), 1–12.
Timothy J. Tautges, Sarah E. Knoop, and Thomas J. Rickmeyer. 2008. Local topological modification of hexahedral meshes. Part I: A set of dual-based operations. *ESAIM Proceedings* 24 (2008), 14–33.
T. Tautges and Sarah E. Knoop. 2003. Topology Modification of Hexahedral Meshes Using Atomic Dual-based Operations. In *Proc. International Meshing Roundtable*.
Timothy J Tautges. 2001. The generation of hexahedral meshes for assembly geometry: survey and progress. *Internat. J. Numer. Methods Engrg.* 50, 12 (2001), 2617–2642.
Timothy J Tautges. 2004. MOAB-SD: integrated structured and unstructured mesh representation. *Engineering With Computers* 20, 3 (2004), 286–293.
Timothy J Tautges, Ted Blacker, and Scott A Mitchell. 1996. The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes. *Internat. J. Numer. Methods Engrg.* 39, 19 (1996), 3327–3349.
Ko-Foa Tchon, Julien Dompierre, and Ricardo Camarero. 2004. Automated refinement of conformal quadrilateral and hexahedral meshes. *Internat. J. Numer. Methods Engrg.* 59, 12 (2004), 1539–1562.
Tessaels. 2022. Tessael. https://www.tessael.com/
W.P. Thurston. 1993. Hexahedral decomposition of polyhedra. *Posting to Newsgroup sci.math* (1993).
Philip M Tuchinsky and Brett W Clark. 1997. The "HexTet" hex-dominant automesher: An interim progress report. In *Proceedings of the 6th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, USA*. 183–193.
Francesco Usai, Marco Livesu, Enrico Puppo, Marco Tarini, and Riccardo Scateni. 2015. Extraction of the Quad Layout of a Triangle Mesh Guided by Its Curve Skeleton. *ACM Trans. Graph.* 35, 1 (2015), 6:1–6:13.
Dimitris Vartziotis and Benjamin Himpel. 2014a. Efficient and Global Optimization-Based Smoothing Methods for Mixed-Volume Meshes. In *Proceedings of International Meshing Roundtable*. 293–311.
D. Vartziotis and B. Himpel. 2014b. Efficient Mesh Optimization Using the Gradient Flow of the Mean Volume. *SIAM J. Numer. Anal.* 52, 2 (2014), 1050–1075.
Dimitris Vartziotis and Manolis Papadrakakis. 2017. Improved GETMe by adaptive mesh smoothing. *Computer Assisted Methods in Engineering and Science* 20, 1 (2017), 55–71.
Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional field synthesis, design, and processing. In *Computer graphics forum*, Vol. 35. Wiley Online Library, 545–572.
Kilian Verhetsel, Jeanne Pellerin, and Jean-Francois Remacle. 2019a. A 44-element mesh of Schneiders' pyramid: Bounding the difficulty of hex-meshing problems. *Computer-Aided Design* 116 (2019), 102735.
Kilian Verhetsel, Jeanne Pellerin, and Jean-François Remacle. 2019b. Finding hexahedrizations for small quadrangulations of the sphere. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.
Ryan Viertel and Braxton Osting. 2019. An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg–Landau theory. *SIAM Journal on Scientific Computing* 41, 1 (2019), A452–A479.
Ryan Viertel, Matthew L Staten, and Franck Ledoux. 2016. *Analysis of Non-Meshable Automatically Generated Frame Fields.* Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
VTK. 2022. File Formats for VTK 4.2. https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf
Ved Vyas and Kenji Shimada. 2009. Tensor-Guided Hex-Dominant Mesh Generation with Targeted All-Hex Regions. In *Proceedings of International Meshing Roundtable*. 377–396.
Erke Wang, Thomas Nelson, and Rainer Rauch. 2004. Back to elements - tetrahedra vs. hexahedra. In *Proceedings of the 2004 International ANSYS Conference*.
Rui Wang, Shuming Gao, Zhihao Zheng, and Jinming Chen. 2018. Hex mesh topological improvement based on frame field and sheet adjustment. *Computer-Aided Design* 103 (2018), 103–117.
Rui Wang, Chun Shen, Jinming Chen, Haiyan Wu, and Shuming Gao. 2017. Sheet operation based block decomposition of solid models for hex meshing. *Computer-Aided Design* 85 (2017), 123 – 137.
Benjamin Weber, Gunilla Kreiss, and Murtazo Nazarov. 2021. Stability analysis of high order methods for the wave equation. *J. Comput. Appl. Math.* (2021), 113900.
Xiaodong Wei, Yongjie Jessica Zhang, Deepesh Toshniwal, Hendrik Speleers, Xin Li, Carla Manni, John A Evans, and Thomas JR Hughes. 2018. Blended B-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 341 (2018), 609–639.
F Weiler, R Schindler, and R Schneiders. 1996. *Automatic geometry-adaptive generation of quadrilateral and hexahedral element meshes for the FEM.* Technical Report. Mississippi State Univ., Mississippi State, MS (United States).
Jean-Christophe Weill and Franck Ledoux. 2019. Towards an automatic and reliable hexahedral meshing.
David R. White, Lai Mingwu, Steven E. Benzley, and Gregory D. Sjaardema. 1995. Automated Hexahedral Mesh Generation by Virtual Decomposition. (1995).
David R. White, Sunil Saigal, and Steven J. Owen. 2004. CCSweep: automatic decomposition of multi-sweep volumes. *Engineering with Computers* 20, 3 (2004), 222–236.
Martin Wicke, Mario Botsch, and Markus H. Gross. 2007. A Finite Element Method on Convex Polyhedra. *Computer Graphics Forum* 26, 3 (2007), 355–364.
Thomas Wilson. 2011. *Simultaneous Untangling and Smoothing of Hexahedral Meshes.* Ph.D. Dissertation. Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports de Barcelona.
T.J. Wilson, J. Sarrate, X. Roca, Rafael Montenegro, and J. Escobar. 2012. Untangling and Smoothing of Quadrilateral and Hexahedral Meshes. *Civil-Comp Proceedings* 100 (2012).
Haiyan Wu and Shuming Gao. 2014. Automatic Swept Volume Decomposition based on Sweep Directions Extraction for Hexahedral Meshing. *Procedia Engineering* 82 (2014), 136 – 148.
Haiyan Wu, Shuming Gao, Rui Wang, and Jinming Chen. 2018. Fuzzy clustering based pseudo-swept volume decomposition for hexahedral meshing. *Computer-Aided Design* 96 (2018), 42–58.
Haiyan Wu, Shuming Gao, Rui Wang, and Mao Ding. 2017. A global approach to multi-axis swept mesh generation. *Procedia engineering* 203 (2017), 414–426.
Jun Wu, Weiming Wang, and Xifeng Gao. 2021. Design and Optimization of Conforming Lattice Structures. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (2021), 43–56.
Shang Xiang and Jianfei Liu. 2018. A 36-Element Solution To Schneiders' Pyramid Hex-Meshing Problem And A Parity-Changing Template For Hex-Mesh Revision. (2018). arXiv:1807.09415 [cs.CG]
Gang Xu, Ran Ling, Yongjie Jessica Zhang, Zhoufang Xiao, Zhongping Ji, and Timon Rabczuk. 2021. Singularity Structure Simplification of Hexahedral Meshes via Weighted Ranking. *Computer-Aided Design* 130 (2021), 102946.
Kaoji Xu and Guoning Chen. 2018. Hexahedral mesh structure visualization and evaluation. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2018), 1173–1182.
Kaoji Xu, Xifeng Gao, and Guoning Chen. 2018. Hexahedral mesh quality improvement via edge-angle optimization. *Computers & Graphics* 70 (2018), 17–27.
Kaoji Xu, Xifeng Gao, Zhigang Deng, and Guoning Chen. 2017. Hexahedral Meshing With Varying Element Sizes. *Computer Graphics Forum* 36, 8 (2017), 540–553.
Soji Yamakawa and Kenji Shimada. 2002. HEXHOOP: Modular Templates for Converting a Hex-Dominant Mesh to an ALL-Hex Mesh. *Engineering with Computers* 18 (2002), 211–228.
Soji Yamakawa and Kenji Shimada. 2003. Increasing the Number and Volume of Hexahedral and Prism Elements in a Hex-Dominant Mesh by Topological Transformations. In *Proceedings of the 12th International Meshing Roundtable*. 403–413.
Soji Yamakawa and Kenji Shimada. 2010. 88-Element solution to Schneiders' pyramid hex-meshing problem. *International Journal for Numerical Methods in Biomedical Engineering* 26, 12 (2010), 1700–1712.
Yang Yang, Xiao-Ming Fu, and Ligang Liu. 2019. Computing Surface PolyCube-Maps by Constrained Voxelization. *Computer Graphics Forum* 38, 7 (2019), 299–309.
A. Egemen Yilmaz and Mustafa Kuzuoglu. 2009. A particle swarm optimization approach for hexahedral mesh smoothing. *Internat. J. Numer. Methods Engrg.* 60, 1 (2009), 55–78.
Yuxuan Yu, Jialei Ginny Liu, and Yongjie Jessica Zhang. 2022. *HexDom: Polycube-Based Hexahedral-Dominant Mesh Generation.* Springer International Publishing, Cham, 137–155. https://doi.org/10.1007/978-3-030-92540-6_7
Yuxuan Yu and Xiaodong Wei. 2020. HexGen and Hex2Spline: polycube-based hexahedral mesh generation and unstructured spline construction for isogeometric analysis framework in LS-DYNA. In *Springer INdAM Serie: Proceedings of INdAM Workshop "Geometric Challenges in Isogeometric Analysis."*.
Pengfei Zhan, Xianhai Meng, Zhongxiang Duan, and Qin Yang. 2018. A Tetra-hex Hybrid Mesh Generation Method Based on Delaunay Triangulation. In *Proceedings of the 2018 International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)*. Atlantis Press, 272–275.
Shangyou Zhang. 2005. Subtetrahedral test for the positive Jacobian of hexahedral elements.
Yongjie Zhang and Chandrajit Bajaj. 2006. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering* 195, 9 (2006), 942 – 960.
Yongjie Zhang, Chandrajit Bajaj, and Guoliang Xu. 2009. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Communications in Numerical Methods in Engineering* 25, 1 (2009), 1–18.
Yongjie Zhang, Yuri Bazilevs, Samrat Goswami, Chandrajit L Bajaj, and Thomas JR Hughes. 2007. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering* 196, 29-30 (2007), 2943–2959.
Y. Zhang, T. J. R. Hughes, and C. L. Bajaj. 2010. An Automatic 3D Mesh Generation Method for Domains with Multiple Materials. *Computer Methods in Applied Mechanics and Engineering* 199, 5-8 (2010), 405–415.

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

Hui Zhao, Xuan Li, Wencheng Wang, Xiaoling Wang, Shaodong Wang, Na Lei, and Xiangfeng Gu. 2019. Polycube Shape Space. *Computer Graphics Forum* 38, 7 (2019), 311–322.

Yao Zheng, Roland W Lewis, and David T Gethin. 1995. FEView: An interactive visualization tool for finite elements. *Finite Elements in Analysis and Design* 19, 4 (1995), 261–294.

Qingnan Zhou. 2022. PyMesh. https://github.com/PyMesh/PyMesh.

Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh arrangements for solid geometry. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–15.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016).

Miloš Zlámal. 1968. On the finite element method. *Numer. Math.* 12, 5 (1968), 394–409.