Optimal Dual Schemes for Adaptive Grid Based Hexmeshing

MARCO LIVESU, CNR IMATI, Italy
LUCA PITZALIS, University of Cagliari and CRS4, Italy
GIANMARCO CHERCHI, University of Cagliari, Italy

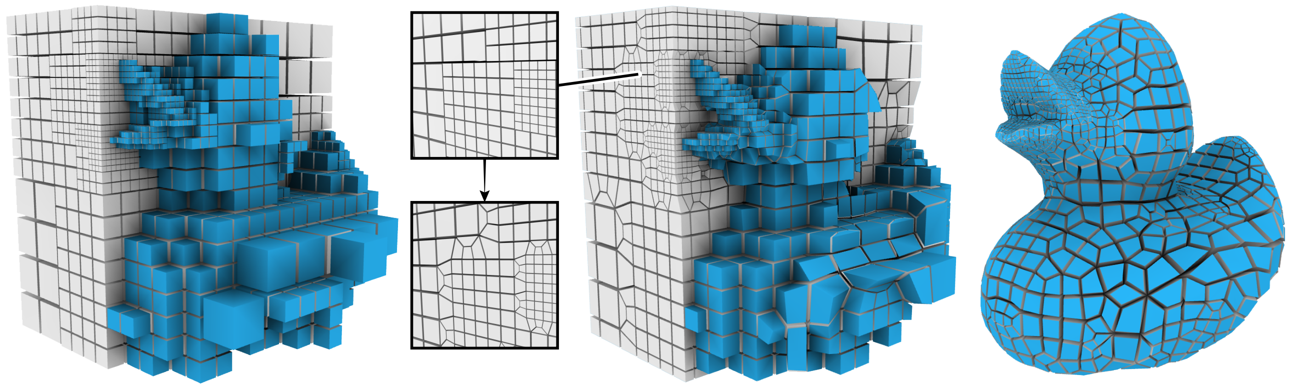

Fig. 1. We propose a novel set of dual schemes to turn an adaptively refined grid into a conforming hexahedral mesh. Compared with prior methods, our schemes allow to process a broader class of input grids, and produce coarser hexahedral meshes with a simpler singular structure.

Hexahedral meshes are an ubiquitous domain for the numerical resolution of partial differential equations. Computing a pure hexahedral mesh from an adaptively refined grid is a prominent approach to automatic hexmeshing, and requires the ability to restore the all hex property around the hanging nodes that arise at the interface between cells having different size. The most advanced tools to accomplish this task are based on mesh dualization. These approaches use topological schemes to regularize the valence of inner vertices and edges, such that dualizing the grid yields a pure hexahedral mesh. In this paper we study in detail the dual approach, and propose four main contributions to it: (i) we enumerate all the possible transitions that dual methods must be able to handle, showing that prior schemes do not natively cover all of them; (ii) we show that schemes are internally asymmetric, therefore not only their construction is ambiguous, but different implementative choices lead to hexahedral meshes with different singular structure; (iii) we explore the combinatorial space of dual schemes, selecting the minimum set that covers all the possible configurations and also yields the simplest singular structure in the output hexmesh; (iv) we enlarge the class of adaptive grids that can be transformed into pure hexahedral meshes, relaxing the tight topological requirements imposed by previous approaches. Our extensive experiments show that our transition schemes consistently outperform prior art in terms of ability to converge to a valid solution, amount and distribution of singular mesh edges, and element count. Last but not least, we publicly release our code and reveal a conspicuous amount of technical details that were overlooked in previous literature, lowering an entry barrier that was hard to overcome for practitioners in the field.

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

Additional Key Words and Phrases: hexahedral meshing, hexmesh, dualization, octree, finite element meshing, mesh generation

Authors' addresses: Marco Livesu, marco.livesu@gmail.com, CNR IMATI, Italy; Luca Pitzalis, University of Cagliari and CRS4, Italy; Gianmarco Cherchi, University of Cagliari, Italy.

1 INTRODUCTION

Volumetric discretizations made of hexahedral cells are ubiquitous in applied sciences, where they are used as computational domains for the numerical resolution of partial differential equations [Schneider et al. 2019; Wang et al. 2004, 2021]. Converting an adaptively refined grid into a pure hexahedral mesh is one of the prominent approaches to hexmeshing, and due to its unbeaten scalability and robustness is the only fully automatic method that successfully transitioned from academic research to industrial software [CoreForm 2020; Dassault Systèmes 2020].

Grid-based methods operate on carefully constructed lattices. When adaptive grids are used, hanging nodes arising at the interface between cells with different size are substituted by dedicated topological schemes that locally restore the all hex connectivity and provide the necessary conforming transitions . For ease of implementation and reproduction, the number of these schemes must be low. At the same time, the scheme set must be rich enough to exhaustively address all the possible configurations, such that convergence to a conforming pure hexahedral mesh is always guaranteed.

Early approaches unsuccessfully tried to directly incorporate hanging nodes in the hexmesh, but this operation is not always possible (Section 2). Maréchal [2009] was the first to observe that if all grid vertices have six incident edges and all edges have four incident cells, then the dual of the grid yields only hexahedral elements. The schemes he proposed operate on small groups of nearby hanging nodes, regularizing their valence (i.e., the number of incident edges) . In this paper, we dive into the details of the dual approach, clarifying some ambiguous aspects of previous methods and also introducing novel topological schemes that overcome the known ones, permitting to obtain – for the same input grids – conforming

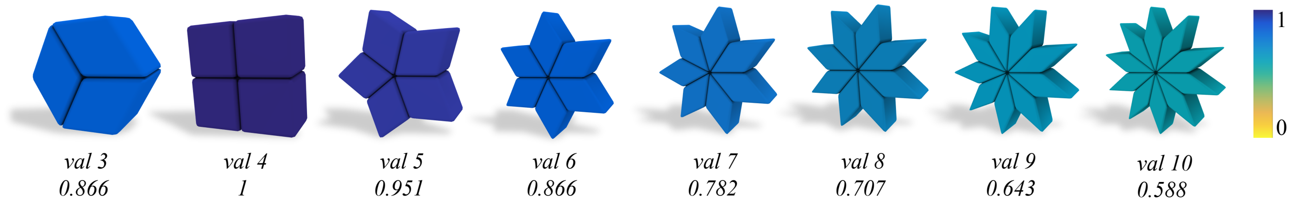

Fig. 2. The number of hexahedra incident to an edge directly impacts the maximum per element quality locally achievable. For each configuration, we show edge valence and the best Minimum Scaled Jacobian that can be obtained from such connectivity . Our schemes always generate edges with valence between 3 and 6. On the dataset released with [Gao et al. 2019], both [Maréchal 2009] and [Gao et al. 2019] introduce singular edges with higher valence.

hexahedral meshes with lower element count and simpler singular structure. Specifically, this article offers four main contributions.

Exhaustivity: we show that the original schemes proposed by Maréchal [2009] and by [Gao et al. 2019; Hu et al. 2013] are not exhaustive, in the sense that they only show the basic transitions, without explicitly deriving the full scheme set that is necessary to handle any possible configuration. Specifically, considering the class of adaptive grids that can be processed with these methods, there exist exactly 20 alternative transitions (Figure 3). As detailed in Section 5, handling all transitions requires a non trivial blending of the known elementary schemes, which was never addressed in previous literature. Our analysis also revealed that previous approaches may occasionally fail to produce a valid mesh (Section 7.2). Conversely, our schemes are guaranteed to always produce the correct result.

Ambiguity: we show that prior schemes are ambiguous, because transitions are internally made of chains of prismatic elements that intersect orthogonally, passing one on top of the other. It follows that schemes are internally asymmetric, and there are always two possible ways to handle an intersection. In flat regions two chains intersect once, hence there are two alternative ways to implement them. Concave edges have three chains that intersect twice, hence there are 2^2 configurations. Concave corners involve three chains that intersect three times, hence there are 2^3 configurations. Interestingly, these choices are often critical, as they may negatively impact the singular structure of the output mesh.

Optimality: we explore the combinatorial space of dual schemes, proving that there are multiple ways to bend a chain of prisms around a concavity and that each method produces a dual hexmesh with different singular structure. We also show that we can avoid high valence edges by wisely selecting the best intersections. In this sense our schemes are *optimal* because, among all the possible chain intersections, they choose the ones that minimize edge valences. Existing tools based on [Gao et al. 2019; Maréchal 2009] unnecessarily insert edges with higher valence, negatively affecting mesh quality (Figure 2).

Weak balancing: we extend the class of adaptive grids that can be transformed into pure hexahedral meshes. Prior schemes require the input grid to be (strongly) balanced, meaning that the difference in the amount of refinement associated with any pair of cells sharing an edge, face or vertex must be lower than 2. We introduce a few additional schemes that permit input grids to obey to a weaker definition of balancing, where only face-adjacent cells must have compatible refinement. This extension allows to greatly reduce the number of mesh elements, introducing up to 64% less hexahedra in the output mesh for same input grid.

Summarizing, this study offers a comprehensive overview – and hopefully a better understanding – of the transition schemes for grid-based hexmeshing, also proposing novel ideas and advancing the field. The schemes proposed in this paper make explicit for the first time the full set of transitions that are necessary to process an adaptively refined grid, and also substantially enlarge the class of input grids that can be converted into a conforming hexahedral mesh. We performed extensive comparative tests on multiple datasets [Gao et al. 2019; Zhou and Jacobson 2016], producing more than 20 thousand hexahedral meshes overall. Based on these results, we can conclude that our schemes are consistently superior than prior art in terms of ability to produce a valid result (i.e., conforming, all-hex), mesh singular structure, and element count (Section 7). To grant maximal diffusion, we publicly share the complete set of schemes and the code necessary to install them. All these contributions have been incorporated into the MIT licensed library CinoLib [Livesu 2019].

2 RELATED WORKS

Grid-based hexahedral meshing was pioneered by Schneiders, who firstly proposed to use regular voxel grids [Schneiders 1996], and soon later introduced adaptively refined grids, obtained through the use of octrees [Schneiders et al. 1996]. Octrees had already been used for adaptive mesh generation, but they were unsuitable to hexmeshing because there were no topological schemes to suppress hanging nodes, and there were no bounds on the topological complexity of each cell [Shephard and Georges 1991].

If adjacent elements in an adaptive grid differ by at most one level of refinement, there exist exactly 2^8 alternative configurations that must be handled. Discarding trivial configurations (i.e., fully refined and unrefined cells) and accounting for symmetries, the number goes down to 20 unique cases [Weiler et al. 1996]. A hexahedralization for four of these configurations was published multiple times [Schneiders 1997, 2000a,b; Schneiders et al. 1996; Tack et al. 1994]. Authors also showed that concave configurations cannot be hexmeshed, because they contain an odd number of quadrilateral faces, a condition for which it is proved that the hexmeshing problem does not admit a valid solution [Mitchell 1996]. Despite being

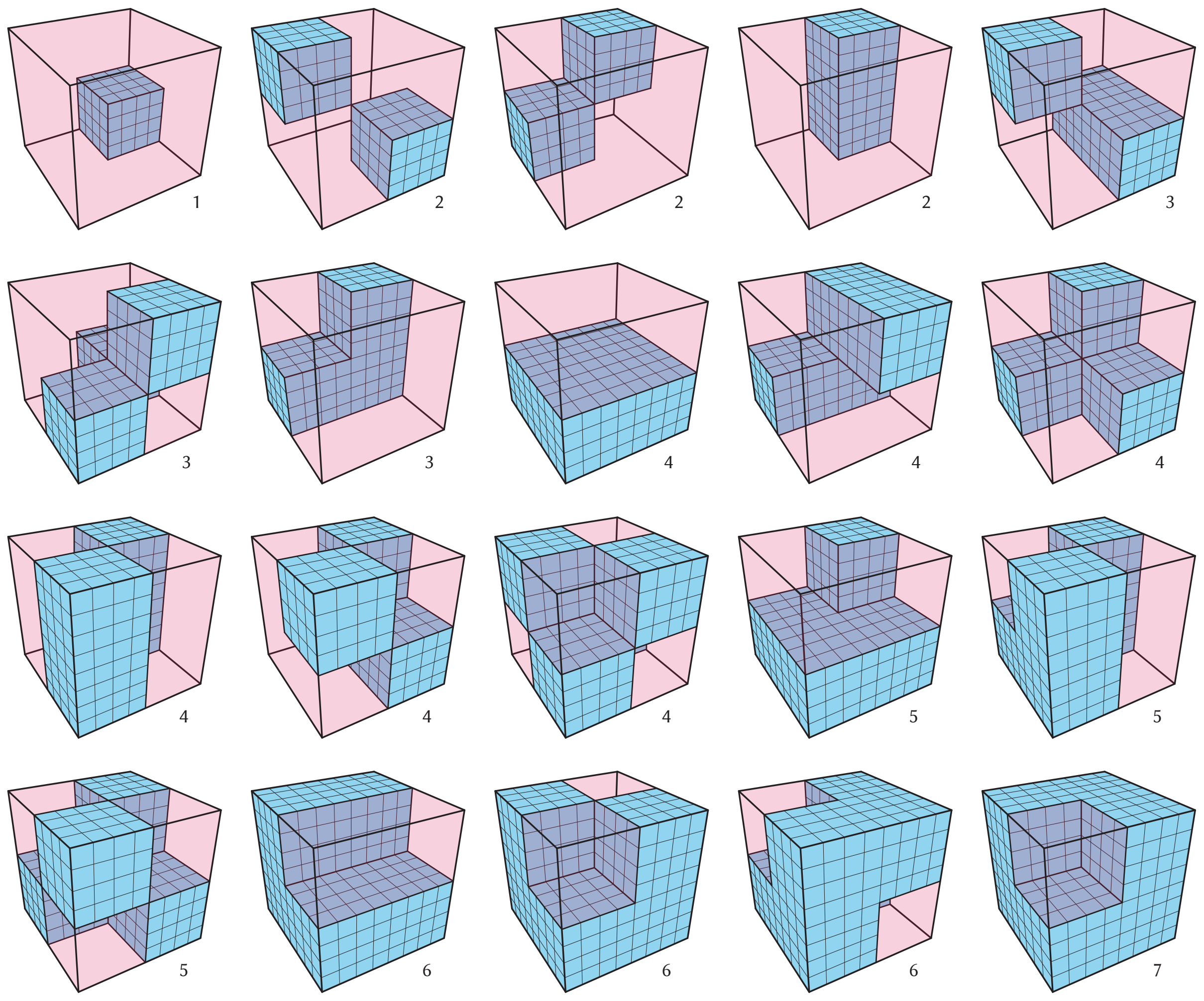

Fig. 3. Exhaustive set of all possible transitions that may arise in an adaptive grid with two alternative levels of refinement. Values next to each cube denote the number of octants filled with a regular 4× 4× 4 sub-grid (blue). The red-shaded empty octants can be imagined filled with a coarser 2× 2× 2 sub-grid, which is not rendered to make the figure easier to parse. These cases can be seen as a volumetric interpretation of the lookup table shown in [Nielson 2004].

non-exhaustive, the known schemes are of practical relevance, and a variety of algorithms used them to generated adaptive hexahedral meshes. Whenever unsupported configurations arise, grid refinement can be iteratively applied to reconduct the problem to the set of known schemes so that a mesh can be created [Zhang and Bajaj 2006]. This allows to potentially mesh any grid, but at the same time is undesirable, because a conspicuous amount of extra refinement may be necessary to converge to a valid solution. Slightly different refinement schemes were proposed in [Ito et al. 2009], but also in this case concave configurations cannot be handled. Conversely, [Livesu et al. 2016] proposed schemes to realize fine-to-coarse transitions along a tubular object, but this approach does not extend to a broader class of shapes.

Considering the impossibility to derive a complete set of hexmeshing schemes, recent literature approached the problem from a different angle. Maréchal [2009; 2016] pioneered the dual approach, which is based on the intuition that cutting grid cells to regularize vertex valences defines a polyhedral mesh that yields a pure hexahedral mesh once dualized. His paper sets the basic ideas but does not document the exhaustive set of schemes that are necessary to handle all the possible transitions described in Figure 3 . Based on the same principles, we independently derived an optimal set of schemes, for which we demonstrate both exhaustiveness and optimality, in the sense that they produce coarser meshes with simpler singular structure.

In recent years a few variations of the dual approach have been proposed. Hu and colleagues [2013] propose to position transition schemes at the inner sides of refined areas, reducing their volumetric extent. The idea is to essentially shift the same schemes one level inwards in the grid, without changing them. Gao and colleagues [2019] proposed to dualize the grid first, and then substitute clusters of non-hexahedral elements with templated all-hex schemes which reproduce patterns very similar to the ones designed by Maréchal. Also these schemes are not exhaustive and, as discussed in Section 7.2, the method implemented in [Gao et al. 2019] may fail to produce a conforming hexahedral mesh. None of these prior methods supports weakly balanced grids, and necessitate to over-refine the input grid to meet more stringent topological criteria for conforming hexmeshing.

Grid-based meshing pipeline. Transition schemes for conforming hexahedral meshing are just a single building block of a more complex pipeline. Typically, the process starts from a coarse regular grid, which is adaptively refined to meet geometric or numerical accuracy. The so generated grid is then further refined to make it topologically suitable for hexahedral meshing [Maréchal 2009]. Transition schemes like ours are then used to secure mesh conformity, and the relevant portion of the grid is extracted, discarding unnecessary cells. Depending on the application, only the interior or the exterior can be retained. For example, the typical goal of FEM analysis is to discover stresses internal to the object, whereas CFD is more concerned with the dynamics happening outside of it (e.g., around the wing of an airplane). The simulation domain is finalized by projecting its boundary onto the target geometry, possibly preserving its feature lines [Gao et al. 2019; Lin et al. 2015]. Since many cuboids may have more than one facet exposed on the surface (or more than one edge participating in a feature line), mesh padding is used to improve mesh topology, ensuring that no element becomes degenerate during the projection [Cherchi et al. 2019]. At the end of this process, the so generated mesh is ready for use and can be coupled with the numerical solver of choice. While this article is fully focused on the topological templates that ensure mesh conformity, each one of the building blocks mentioned above has a dedicated line of research. Our schemes are compatible with any existing grid-based meshing pipeline.

Other pipelines. Hexahedral meshing is a vast topic, and a variety of alternative techniques have been proposed in literature, such as polycubes [Fang et al. 2016; Fu et al. 2016; Gregson et al. 2011; Huang et al. 2014; Livesu et al. 2013], advancing front methods [Kremer et al. 2014], sweeping methods [Gao et al. 2015], and methods that align to some guiding field [Corman and Crane 2019; Li et al. 2012; Liu et al. 2018; Livesu et al. 2020; Solomon et al. 2017]. Most of these methods are notoriously superior than grid methods, in the sense that they produce meshes with much simpler singular structure and often much higher per element quality. Nevertheless, none of these algorithms can be compared with grid-based approaches in terms of robustness and scalability, making the use of grids the only feasible solution to reliably process large collections of shapes of any size and complexity.

3 DUAL MESHING: CONSTRAINTS AND DESIDERATA

The dual idea is a broad topological concept, with applications in many scientific fields. In mesh generation dualization has been widely used, e.g. to transform a Voronoi diagram into a simplicial mesh [Lévy and Liu 2010], or to generate quadrilateral [Campen et al. 2012; Nielson 2004] and hexahedral [Gao et al. 2019; Hu et al. 2013; Maréchal 2009; Tautges and Knoop 2003] meshes.

Considering a (primal) cellular complex composed of V vertices, E edges, F faces and C cells, its dual mesh is a cellular complex having:

- one vertex for each primal cell $c \in C$
- one edge for each primal face $f \in F$
- one face for each primal edge $e \in E$
- one cell for each primal vertex $v \in V$

In particular, the valence of each dual vertex corresponds to the number of faces of its associated primal cell. The valence of each dual edge corresponds to the number of sides of its primal face. The number of sides of each dual face corresponds to the valence of its associated primal edge. The number of faces of each dual cell corresponds to the valence of its associated primal vertex.

From a topological perspective, a hexahedron is a solid with 8 vertices, 12 edges, and 6 quadrilateral faces. Considering the definition above, one can always generate a pure hexahedral mesh via dualization if and only if:

- each primal vertex has valence 6, because its associated dual cell has 6 faces
- each primal edge has valence 4, because its associated dual face will be a quad

In addition to these strict topological requirements, it is practically relevant to ensure that the so generated hexmesh has a good singular structure, meaning that it locally resembles a regular grid almost everywhere. To this end, it is desirable that the majority of inner dual vertices have valence 6, and that the majority of dual edges have valence 4. Thinking about these properties in terms of their relation with the primal mesh, it turns out that a good adaptive grid should have as many cells as possible composed of 6 faces, and as many 4-sided faces as possible. In particular, it is important to avoid primal faces with many sides, because they produce high valence singular edges in the dual hexmesh, which negatively impact per element quality (Figure 2). The schemes proposed in this paper are designed to fully address topological constraints, and also to optimize the fulfillment of practical desiderata.

4 BASIC TRANSITIONS

In this section we start from the basic scheme originally proposed in [Maréchal 2009] and show how it can be adapted to convex, concave, and corner configurations. As it will become clear in the remainder of the section, there are multiple ways to perform this task. We will exhaustively show all the possible versions and select the ones that are optimal with respect to the desiderata expressed in Section 3.

The core idea is identical to the 2D case depicted in Figure 5, where the two hanging nodes are suppressed by forming two triangles connected through the vertical edge in between them. However, the 3D realization is more convoluted, because any non conforming transition between a 4×4 and a 2×2 grid generates 14 hanging vertices with valence 5. Following the analogy with the 2D case, we can imagine to extrude the triangles that suppress the hanging nodes,

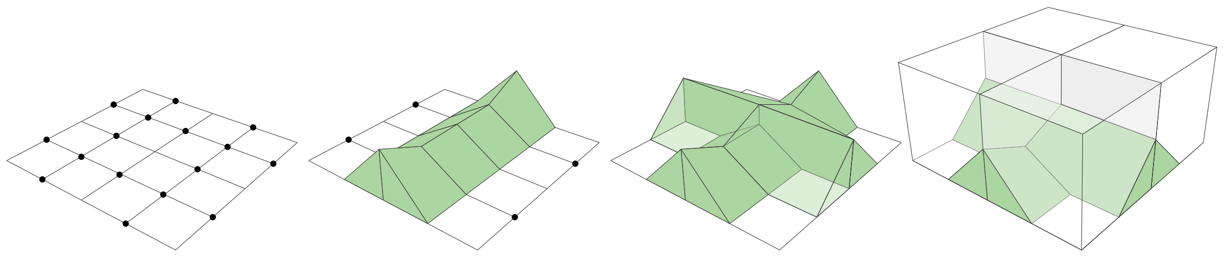

Fig. 4. Basic scheme to transition from a flat 4×4 to a 2×2 grid. From left to right: there are 14 hanging nodes with valence 5 (black dots). The first chain of prismatic elements with triangular cross section suppresses all hanging nodes but four. The second chain intersects the first one orthogonally, and secures valence 6 for all vertices. The four upper cells reproduce the 2×2 structure, completing the transition. Some of the faces are transparent to better inspect the interior topology.

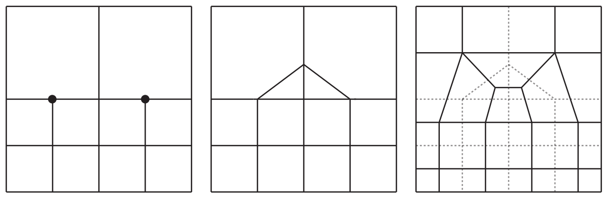

Fig. 5. A 2D example of the dual approach proposed in [Maréchal 2009]. Left: an adaptively refined grid has two hanging vertices (black dots) at the interface between elements of different sizes. Middle: connecting them through the vertical edge in between ensures that all internal nodes have valence four. Right: dualizing the grid yields a pure quadrilateral mesh.

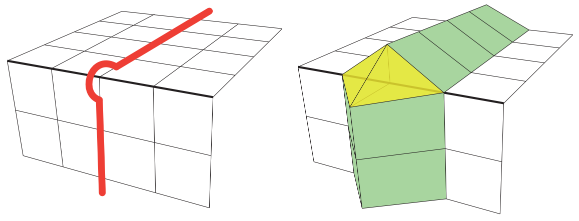

Fig. 6. When a chain of prismatic elements turns 90° to traverse a convex edge, two tetrahedral elements (yellow) are necessary to adjust mesh topology and provide the necessary bending.

which become chains of prismatic elements with triangular cross-section. Each transition requires two such chains, that intersect orthogonally at the middle of the grid. If the chains were identical, their intersection would define a valence 8 vertex, violating the constraints expressed in Section 3 and thus failing to produce a pure hexahedral dual mesh. To keep vertex valences under control, the trick is to make sure that the two chains do not intersect at the same height, but rather one passes below the other, splitting the valence 8 vertex into two valence 6 vertices. Consequently, any time there is an intersection, one must choose which of the two chains passes below the other, leading to ambiguity. Figure 4 shows how the topology of the two chains must be arranged to secure the correct vertex and edge valences. One could alternatively choose to have the upper chain passing below the lower one. This choice is completely harmless if the intersection occurs at a flat region, because the global amount and type of primal mesh elements does not change.

Given this basic scheme, the whole idea behind dual hexmeshing is to suppress all hanging nodes by designing a network of prismatic chains that wind around clusters of grid elements having the same amount of refinement. Since each cluster is a regular sub-grid, its outer surface is also regular, therefore chains always intersect pairwise in an orthogonal manner. In practice, this means that all we need is to be able to adapt the scheme in Figure 4 to allow these chains to turn at the convexities and concavities of each refined cluster.

4.1 Convex transitions

Two chains of prisms that meet at the convex edge of a refined area can be welded together by using two tetrahedral elements that form a bridge between the cross sections of the incoming chains (Figure 6). Differently from flat and concave transitions, this scheme is not ambiguous because no intersections between orthogonal chains are involved.

4.2 Concave transitions

Transitions across concave edges are more complex, because four different chains of prismatic elements are involved. Two of them are parallel to the concave edge, and are positioned aside from it. The other two are orthogonal to the concave edge, and are the ones that need to be merged into a single chain that turns at the concavity. These four chains intersect pairwise at the left and right of the concave edge. Depending on how these intersections are realized, the transition changes. There are two different ways two handle each intersection (one chain goes below, one above), therefore there exist 2^2 alternative solutions. Ignoring symmetries, the amount of unique schemes reduces to three. Specifically, if the two chains that merge at the concave edge pass both below their respective intersections, their blending can be realized using three pentagonal faces (Figure 7, left). If one of the two chains passes above its intersection, then three hexagonal faces are needed (Figure 7, middle). Finally, if both chains pass above their intersections, three heptagonal faces are needed (Figure 7, right).

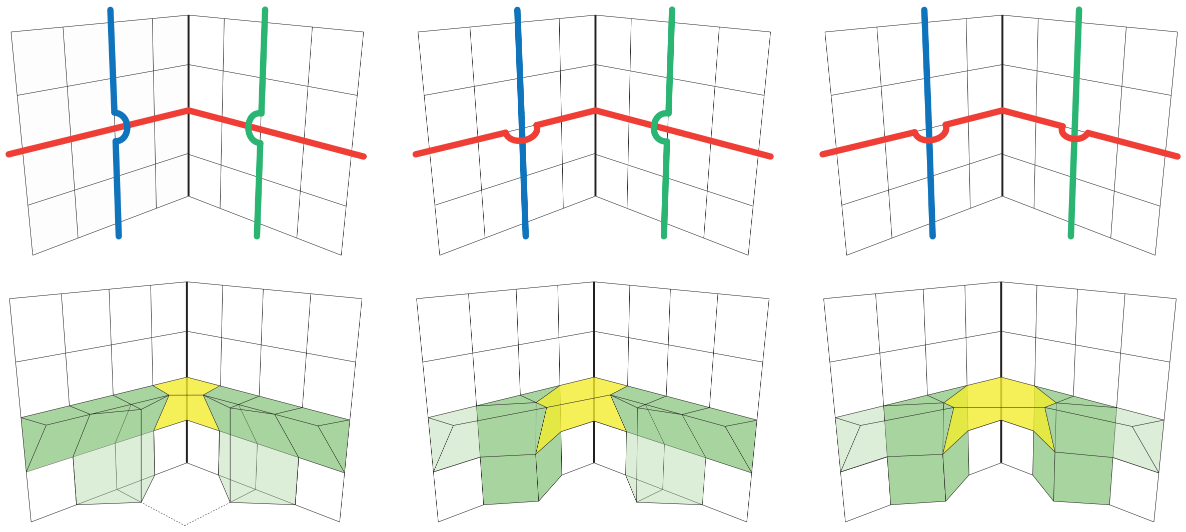

Fig. 7. There are three different ways to bend a chain of prismatic elements along a concave edge. Left: if the chain passes below both intersections aside the concavity, the bending can be realized with three pentagonal faces. Middle: if the chain passes below one intersection and above the other, the bending necessitates hexagonal faces. Right: if the chain passes above both intersections, heptagonal faces are needed. Transition elements are highlighetd in yellow. Note that in all cases two hexagonal faces are needed to handle the bottom and top corner faces (see the bottom left dashed lines). The leftmost solution is the optimal one, because it introduces the least amount of high valence irregular edges in the dual hexmesh.

Recalling that primal faces become edges in the dual mesh, and that the valence of such edges correspond with the number of sides of their primal face, we can conclude that – depending on the scheme of choice – concavities may introduce irregular edges with valence 5, 6 or 7. In order to optimize the criteria expressed in Section 3 we always adopt the transition that produces valence 5 edges, obtaining the simplest singular structure in the output mesh. Note that regardless of the configuration of choice, the top and bottom lids of a concave edge are essentially two quads with two corners cut (to account for the incoming chains). This means that the full scheme will still produce two valence 6 edges in the dual mesh (see the dashed lines at the bottom left of Figure 7). Nevertheless, our choice minimizes the extent of high valence singularities in the output hexmesh, completely avoiding valence 7 edges and reducing the amount of valence 6 singular edges to only two.

4.3 Transitions around corners

Prismatic chains never traverse the corners of a cluster of refined elements directly, but each corner has three chains that wind around it and mutually intersect each other three times. If the corner is convex, these intersections are handled with the flat scheme in Figure 4, and always produce a mesh with equivalent singular structure. Conversely, concavities require to use a blending of the schemes for concave edges shown in Figure 7. Differently from a single concavity, which can always be handled with the simplest among the three possible options, concave corners are the meeting point of three mutually orthogonal concave edges. The interplay between the chains winding around the corner is such that it becomes impossible to make sure that each chain passes below all intersections it is involved in. More precisely, three mutual intersections and two alternative ways to handle each one of them define a combinatorial space of 2^3 alternative solutions. Removing the symmetries, there exist only two ways to handle a concave corner: in one case, each concave turn involves a chain that passes above one intersection and below the next one (Figure 8, left). In the other case, all the three transitions shown in Figure 7 arise. Also in this case, our preference goes to the left configuration because it fully avoids the generation of singular edges with valence 7 in the output hexmesh and minimizes the amount of valence 6 edges.

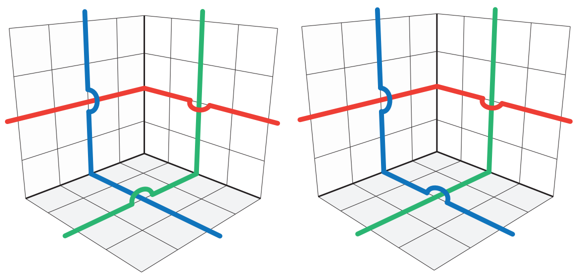

Fig. 8. Corners of a refined area always have three chains of prisms winding around them. Since each intersection chain can be arranged in two different ways, there are 2^3 alternative configurations, which removing symmetries reduce to the two shown in this figure. The left configuration is fully symmetric, as the chain traversing each concave edge passes above one intersection and below the other one. The right configuration exposes all the three possible cases (one chain fully below, one chain fully above, and one chain both below and above). The leftmost configuration is better because it only introduces valence 6 edges in the hexmesh, whereas the one at the right also introduces valence 7 singular edges.

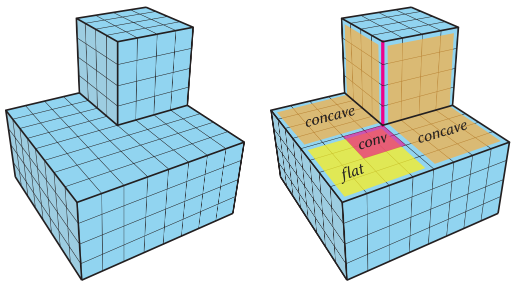

Fig. 9. A non trivial transition involving a flat area, two concave edges, and one convex edge. The basis of the flat and convex schemes conflict (right), therefore basic transitions cannot be used directly, but must be combined in order to produce hybrid schemes that adapt to the local shape of the grid.

5 SCHEMES

The basic transitions shown in the previous section cannot be directly used to transform an adaptively refined grid into a pure hexahedral mesh. As shown in Figure 9, many local configurations will necessitate hybrid transitions, which are a blend of the atomic patterns designed for the flat, convex and concave cases.

To enumerate all the possible local configurations that may arise, we can imagine working in a grid where only two alternative refinement levels are possible: coarse and fine. Note that this hypothesis is not restricting: all grid-based methods assume that the adaptive grid is *balanced*, which means that any couple of face, edge, or vertex adjacent cells can differ by at most one level of refinement [Maréchal 2009]. It follows that indeed – at a local level – only two levels of refinement are possible.

Let us imagine having a cube split into 8 octants, and filling each octant either with a coarse $2\times 2\times 2$ or with a finer $4\times 4\times 4$ sub-grid. Since for each octant there are two possible choices, there exist 2^8 alternative assignments. Ignoring symmetries and removing the two fully regular grids obtained by filling all octants with the same element, we obtain a set of 20 unique configurations, which correspond to all the possible transitions that may arise in a balanced adaptive grid (Figure 3). This combinatorial space is equivalent to the one explored by primal approaches (Section 2), and is also equivalent to that of Marching Cubes [Lorensen and Cline 1987], which associates a sign to the cube's corners and obtains the same 20 possible binary assignments. This relation is even clearer in the dual version of MC [Nielson 2004], which shows a lookup table that, up to a volumetric interpretation, is equivalent to ours. Similar surface schemes had already been introduced in the Cuberille algorithm [Chen et al. 1985; Herman and Liu 1979].

Implementing a transition scheme for each of the 20 patterns in Figure 3 is, therefore, the simplest way to have a lookup table that exhaustively addresses all the cases. Note that this number is much bigger than the three schemes often presented as exhaustive in previous literature, and the reason is that prior works did not present the actual schemes, but just a particular instance of the basic transitions in Section 4.

To reduce the amount of configurations to the minimum, we present here an alternative method we used to encode the patterns. The basic idea is that many of the 20 schemes share some component, sometimes exactly as it is, some other times up to a rotational and reflection degree of freedom. We exploit this redundancy to define a minimum set of 8 atomic elements, which never overlap, and can be grouped together to reproduce all the 20 possible transitions. The full set is depicted in Figure 10 and comprises: one flat element (F); one hybrid flat and convex element (F+C); three convex elements (C1, C2, C3) that handle 1, 2 or 3 prismatic chains incident to the same grid cell; one element for concave edges (E), and two elements for concave vertices, once for the center (VC) and one for its sides (VS). The basic transitions discussed in Section 4 can be reproduced by considering simple arrangements of these 8 elements. As an example, the flat scheme in Figure 4 is composed of four elements of type F which can be positioned by starting from one of them and reflecting it four times across one of its lateral faces. Similarly, all the transitions shown in Figure 3 can be obtained by compositions of the same 8 atomic elements.

A pictorial illustration of the installation process is shown in Figure 11. Note that the sequence of operations is not mandatory. Since these atomic blocks do not conflict with each other, one can start by positioning a single brick, and simply proceed by placing the subsequent ones so as to preserve mesh conformity, always obtaining the same result.

6 WEAKLY BALANCED GRIDS

All known methods for grid-based adaptive hexmeshing require that the input grid is *strongly* balanced, which means that cells that differ by more than one level of refinement must not share any vertex, edge, or face. In this section we discuss a minimal extension of our basic schemes, which allows to relax this stringent requirement, embracing a much broader class of input grids and ultimately permitting us to obtain coarser hexahedral meshes that fully preserve the input prescribed refinement .

Our key observation is that when there is high disparity in the refinement associated to nearby cells, satisfying the strong balancing criterion requires a conspicuous amount of additional refinement, significantly increasing the cell count. Conversely, if the balancing criterion was *weaker*, meaning that restrictions applied only to face-adjacent cells, the amount of necessary subdivisions would be much lower (Figure 13). From a combinatorial point of view, the extension to weak balancing opens conforming hexahedral meshing to a much wider set of adaptively refined grids. Following the analogy with the cube example in Section 5, one can enumerate all possible configurations by considering a cube split into 8 octants, associating to each octant either a $2\times2\times2$, a $4\times4\times4$, or a $8\times8\times8$ sub-grid. Since there are three alternative choices for octant refinement, there exist $3^8 = 6561$ possible configurations. Discarding all configurations that are identical up to a rotation, the number of unique configurations goes down to 332. Furthermore, discarding all configurations that violate weak balancing (i.e., all grids where octants with $2\times2\times2$ and $8\times8\times8$ refinement are face-adjacent), the number of remaining unique configurations that must be handled is 58.

Weakly balanced grids may contain edges shared between cells with three different levels of refinement, and vertices incident to cells spanning four different levels of refinement. Luckily, the vertex

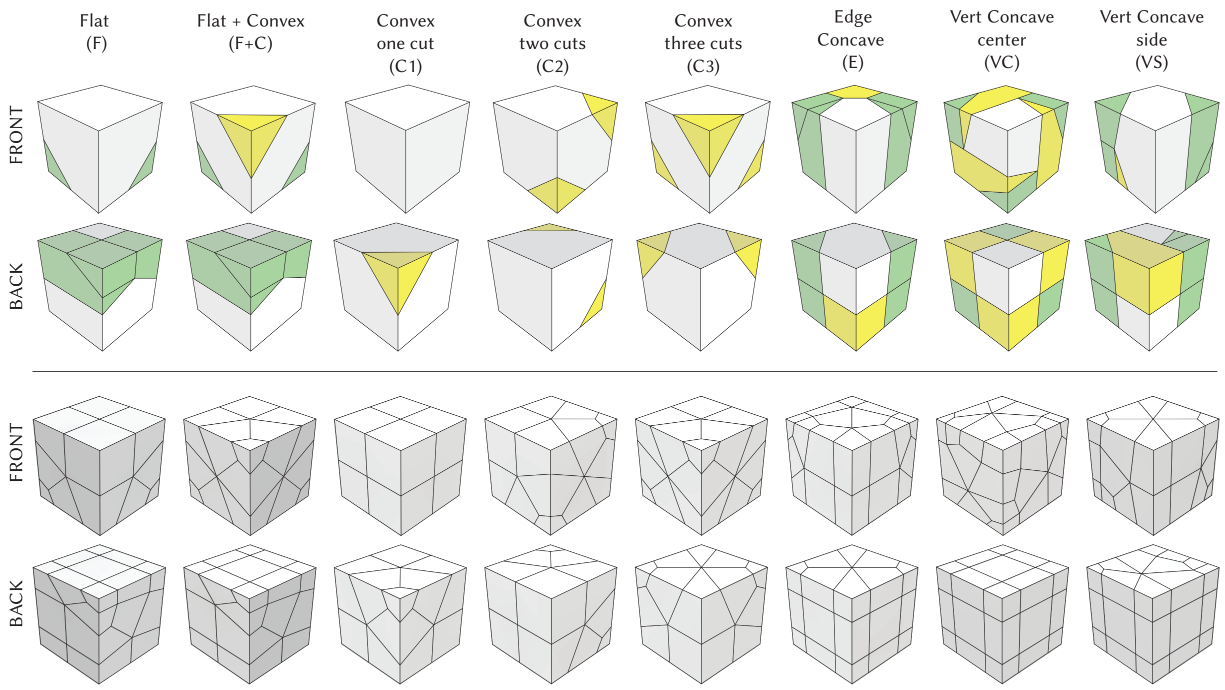

Fig. 10. The 8 atomic schemes used to mesh all the transitions listed in Figure 3. Top: elements are color-coded with respect to their type. Green elements belong to prismatic chains that suppress the hanging nodes of a refined cluster. Yellow elements allow the green chains to bend around convex and concave edges. White elements are lids that fill the remaining volume. Bottom: hexahedralized transitions obtained with standard mesh dualization.

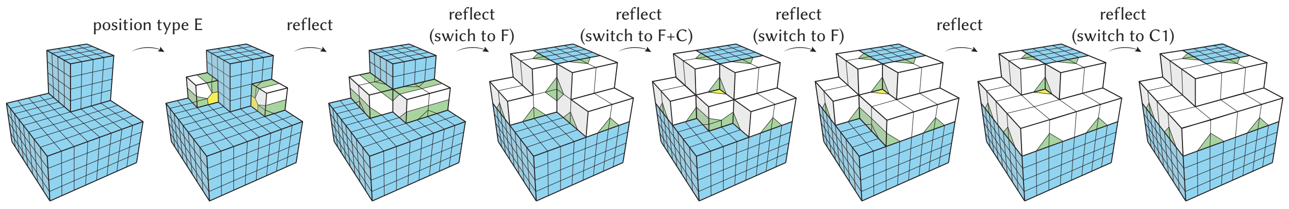

Fig. 11. Example of installation of the atomic schemes in Figure 10 for a complex case involving flat, convex, and concave regions. All the necessary transitions can be realized with a combination of rigid movements and reflections of the basic schemes. The installing sequence is not mandatory, and any alternative sequence would provide the same result.

case does not require any special handling because – regardless of the size disparity – any grid vertex has 8 incident cells and 6 incident edges, which means that it always yields a hexahedron in the dual mesh. This is also the reason why weakly balanced grids in 2D do not necessitate dedicated schemes. Conversely, edges shared by cells spanning three levels of refinement generate hanging vertices that must be incorporated into the mesh connectivity. As shown in Figure 12 there are four possible cases, which correspond to an open concave edge, or a concave corner where 1, 2 or 3 of the incident concave edges contain additional hanging vertices.

It is interesting to notice that the concave edges where the additional hanging nodes arise are convex edges for the (twice more refined) grid minors opposite to the concavity. This means that the schemes we need are essentially a blend between the basic convex and concave schemes shown in Figure 10. A pictorial illustration of how to realize this blend for open concave edges is shown in Figure 14. Note that the tetrahedra that realize the convex transition are located across the polygonal faces that permit the concave bending, transforming them from n−gons to $(n + 1)$−gons. Specifically, the transition for concave edges required the use of pentagonal faces, which now become hexagons. The transition for concave corners required the use of hexagons, which now become heptagons. In terms of output results, this means that weakly balanced grids can be transformed into pure hexahedral meshes with singular edges of valence 3, 5, 6, and 7 using the 8 schemes in Figure 10, plus 5 additional schemes shown in Figure 15.

7 DISCUSSION

We implemented the whole scheme set and the software necessary to process an input grid in C++, releasing the code within

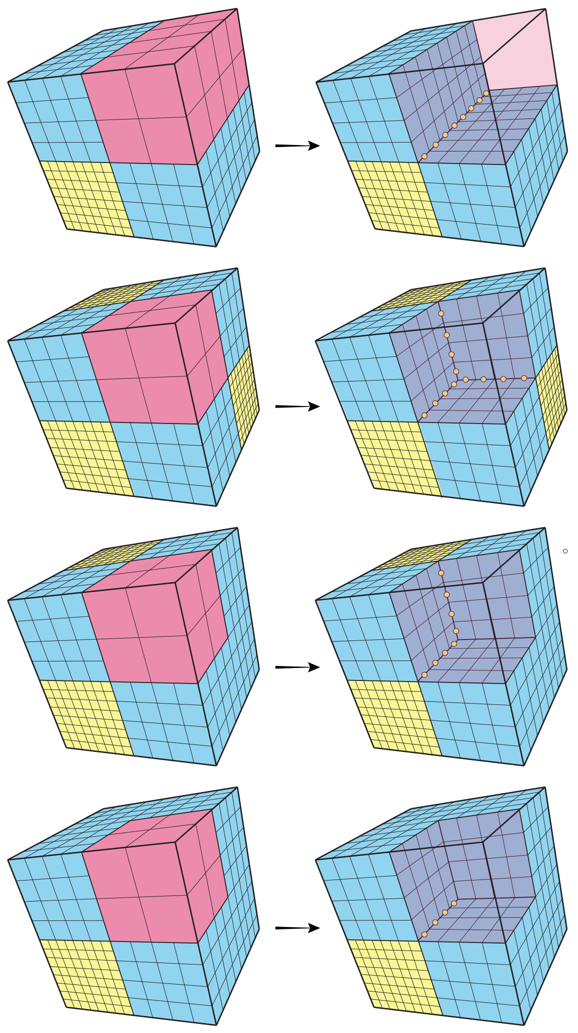

Fig. 12. Weakly balanced grids may exhibit configurations where cells with three different levels of refinement are incident to the same edges, generating new hanging vertices that cannot be suppressed with prior schemes (yellow dots). Each row shows one of the four possible cases: elements can all be incident to the same concave open line, or to all (or a subset) of the three concave lines that terminate in a concave corner. As can be noticed hanging vertices belong to the finest sub-grids, and their suppression demands a blend between a convex transition (for the yellow part) and a concave transition (for the blue part).

unbalanced				
0	0	0	0	0
0	0	0	0	0
0	0	5	0	0
0	0	0	0	0
0	0	0	0	0

strongly balanced				
3	3	3	3	3
3	4	4	4	3
3	4	5	4	3
3	4	4	4	3
3	3	3	3	3

weakly balanced				
1	2	3	2	1
2	3	4	3	2
3	4	5	4	3
2	3	4	3	2
1	2	3	2	1

Fig. 13. Starting from an unbalanced grid (left), fulfilling *strong balancing* demands 80 steps of extra refinement (middle). If *weak balancing* is permitted, the amount of necessary refinement is reduced by 25%.

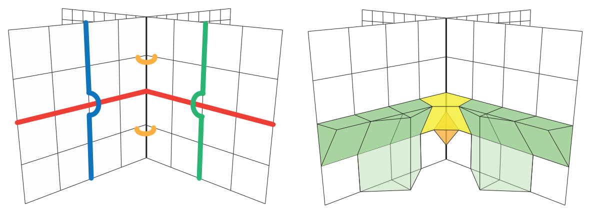

Fig. 14. Left: hybrid convex/concave transition involving three different levels of refinement. Besides the three canonical chains of prisms of a standard concavity (in red, blue, green), there are two extra chains that take a convex turn around the concave edge (orange). Right: the two tetrahedral elements that ensure the convex transition (orange) partially overlap with the concave transition (yellow). As a result, the yellow faces – that were pentagons in the basic concave transition – become hexagons.

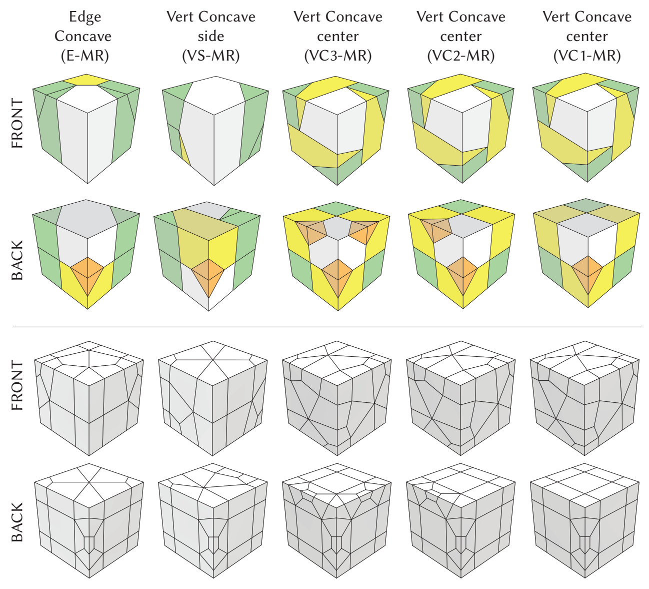
Edge Concave (E-MR)
Vert Concave side (VS-MR)
Vert Concave center (VC3-MR)
Vert Concave center (VC2-MR)
Vert Concave center (VC1-MR)
FRONT
BACK
FRONT
BACK

Fig. 15. Hybrid concave/convex schemes to handle weakly balanced grid having cells with three levels of refinement incident to the same grid edges. Top: green elements are pieces of the prismatic chains. Yellow elements allow concave bending. Orange elements allow convex bending. White cells are lids to complete the volume. Bottom: hexahedralized transition blocks obtained with standard mesh dualization.

the MIT licensed library CinoLib [Livesu 2019]. Specifically, the 8+5 atomic elements whose combination realizes all the possible schemes for strongly and weakly balanced grids are hardcoded as general polyhedral meshes. Each such element can be installed in various alternative orientations, because the octahedral group O contains 24 rotations [Nieser et al. 2011; Solomon et al. 2017]. The code we released for installation is designed to translate, rotate, reflect and scale the atomic blocks, reproducing any desired transition. The so generated polyhedral meshes can be readily transformed

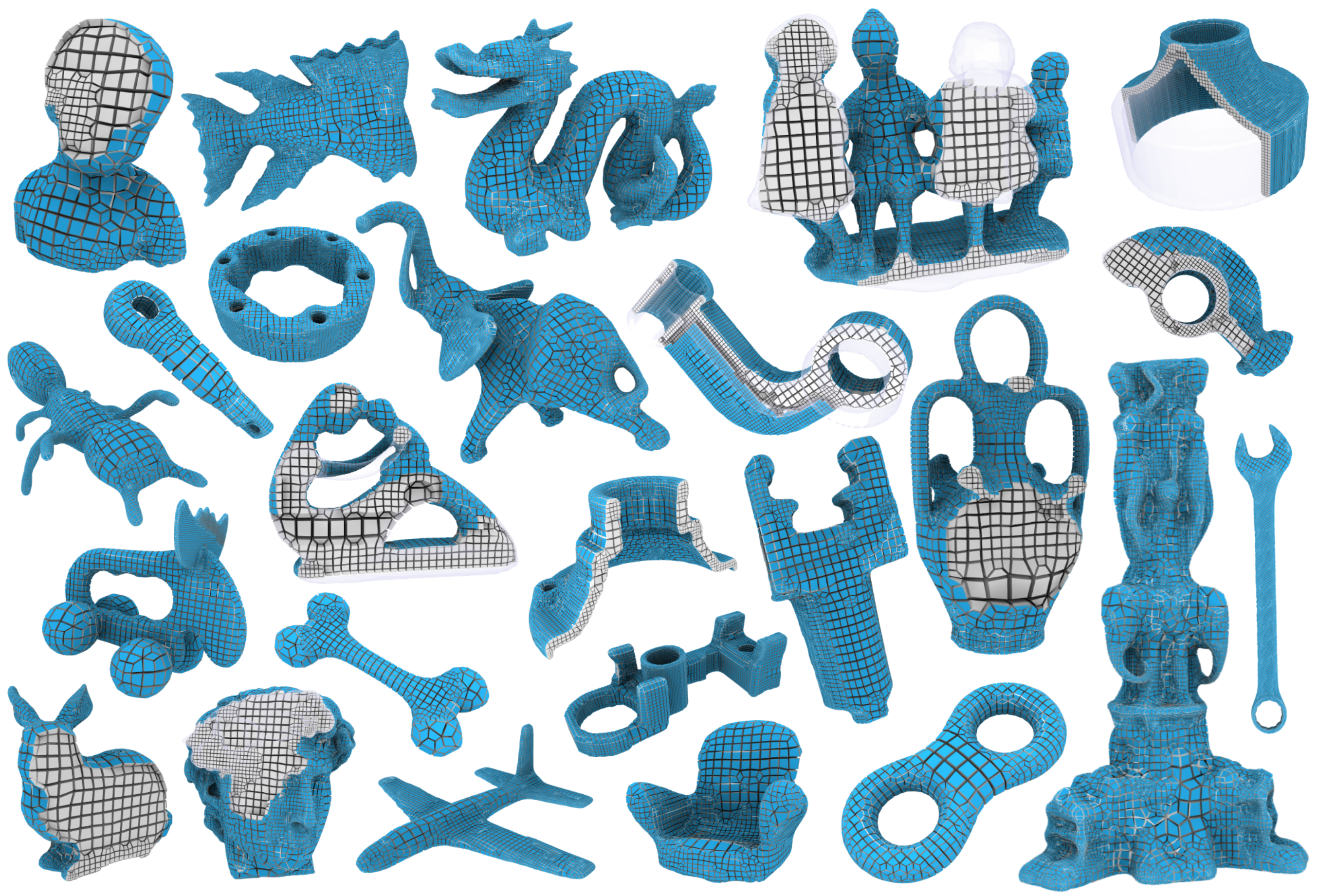

Fig. 16. A random subset of models used to validate our method. To produce these meshes we retained only the subset of grid elements completely internal to the input shape, and projected and smoothed the boundary vertices using [Livesu et al. 2015].

into pure hexahedral meshes with standard mesh dualization, which is also available in the same library.

7.1 Validation

To validate our schemes, we implemented a classical grid-based meshing pipeline as described in Section 2. Given an input shape, we initialized an empty octree covering its bounding box, and then iteratively split octants intersected by the input surface until the grid size was at least twice as big as the local thickness of the shape, measured with the SDF [Shapira et al. 2008]. We then applied additional refinement to satisfy the topological criteria necessary for processing, and eventually installed the transition schemes described in Sections 5 and 6, applying mesh dualization to produce the output meshes. With this pipeline, we batch processed the dataset released with [Gao et al. 2019], which comprises 202 organic and CAD models, and the clean version of the Thingi10K [Zhou and Jacobson 2016] dataset, released by the authors of [Hu et al. 2018]. In all cases, our method was able to successfully produce a conforming hexahedral mesh. A mosaic of hexahedral meshes produced with our method and projected and smoothed with [Livesu et al. 2015] is shown in Figure 16. Note that the focus of this work is purely on the transition schemes, and this is just a simplistic workflow that offers no guarantees in terms of mesh quality and geometric fidelity. As reported in Section 2 scientific literature offers various alternative choices for octree splitting rules, padding, feature preservation, and robust surface projection. Our approach can be combined with any of the existing techniques to obtain a fully-fledged meshing pipeline.

7.2 Comparisons with prior art

We provide both direct and indirect comparisons with prior art. Our natural competitors are the original approach proposed by Maréchal in [2009] and the alternative set of schemes recently proposed in [Gao et al. 2019]. We considered the ability to converge to a valid solution (i.e., conforming, all-hex), the amount and distribution of singular edges, and the mesh size. Our method provides advantages with respect to all these criteria, as detailed in the remainder of this section.

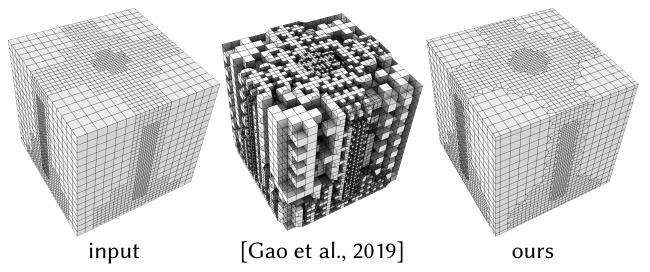

Fig. 17. A failure case for the dual method proposed in [Gao et al. 2019]. In this case, their algorithm is not able to produce a conforming hexmesh starting from an adaptive grid that is fully compliant with the minimum requirements for hexahedral meshing. Overall, testing their reference implementation on the Thingi10K dataset, we isolated 37 similar failures. All adaptive grids were created with the reference implementation released by the authors. On the same grids, our method was always able to produce a valid mesh with the 8 schemes for strongly balanced grids proposed in Section 5. Input and outputs grids are included in the additional material for all failure cases.

Type	ID	F_3	F_4	F_5	F_6	F_7
Flat	F	5	10	5	-	-
Flat + Convex	F+C	9	9	4	2	-
Convex 1 cut	C1	4	3	3	-	-
Convex 2 cuts	C2	8	1	4	1	-
Convex 3 cuts	C3	12	-	3	3	-
Concave edge	E	8	14	7	1	-
Concave vertex central	VC	9	21	-	6	-
Concave vertex side	VS	8	16	3	3	-
Concave edge	E-MR	14	10	10	2	-
Concave vertex central	VC1-MR	16	17	4	5	1
Concave vertex central	VC2-MR	23	14	6	5	2
Concave vertex central	VC3-MR	30	12	6	6	3
Concave vertex side	VS-MR	15	12	7	2	1

Table 1. Topological details for all the transition schemes proposed in the article. For each scheme we report the number of polygonal faces they contain. Once dualized, each polygon translates to an edge in the hexmesh with valence corresponding to the number of sides in the primal (the subscript i in the F_i notation).

Setup. We performed a direct comparison using the original source code released by the authors of [Gao et al. 2019]. Their software realizes a complete hexmeshing pipeline, interleaving grid refinement with surface projection, progressively increasing mesh density until a target geometric accuracy is reached. Since our contributions are purely on the topological step of the pipeline, we isolated from their code the portions relative to adaptive grid generation, balancing and dual scheme installation, fixing the octree depth refinement in the static range [4,7]. For each input model, we run their code to produce a conforming hexahedral grid, and also dumped on a separate file the same grid prior to balancing and scheme installation, which we loaded in our software and processed with our schemes.

Failures. Provided an adaptive input grid that fulfills all topological requirements (i.e. balancing and pairing), the first and foremost property of a set of transition schemes is its ability to produce a valid output mesh that contains only hexahedral cells and is conforming. As detailed in Sections 5 and 6 our schemes fully cover the combinatorial space of patterns for strongly and weakly balanced grids, and are therefore guaranteed to always produce a correct mesh. This was also empirically verified by processing our testing datasets multiple times with varying settings (e.g., for balancing and grid refinement), producing more than 20 thousand hexahedral meshes overall.

The schemes proposed in [Gao et al. 2019] permit to substitute hanging nodes with a *frustum* (Figure 4 in their paper) and also provide three topological bridges to account for adjacent frustums that form a flat, convex, and concave open-angle (Figure 5 in their paper). Concave corners are not taken into account, as well as conflicts that arise between basic schemes (see, e.g., Figure 9). Restricting to the four schemes they present, only 7 out of 20 possible cases shown in Figure 3 can be handled. In their code, the authors complement their schemes with a heuristic approach, which locally modifies the mesh connectivity in order to ensure mesh conformity. This approach is not documented in the article and, despite the software being able to produce a conforming mesh in most of the cases, we isolated 37 failure cases in our testing dataset. Figure 17 shows a typical failure case. All the other failures can be found in the attached material, together with the associated input grids and our corresponding valid solutions. Also [Maréchal 2009] does not propose an exhaustive set of schemes. The article describes how to construct a flat transition that is identical to ours (Figure 4), but it did not provide details on how this can be modified to enable bending and winding around concave edges and corners. To our knowledge, none of the authors of prior articles were ever able to reproduce the schemes in [Maréchal 2009], and comparisons were always based on a one-month trial of the commercial software implementing this method [Dassault Systèmes 2020]. The tool – formerly called MeshGems – has been recently acquired by another group, and we were not able to obtain an evaluation copy. An exhaustive set of schemes was likely devised for this commercial tool, but we could not verify our assumption due to the lack of reference software.

Singular Structure. Regarding the impact that transition schemes have in the singular structure of the hexahedral mesh, in Table 1 we detail the number of sides for each polygonal primal face. As mentioned in Section 3, primal faces with n sides transform into edges with n incident hexahedra in the dual. Therefore, once dualized, strongly balanced grids contain only singular edges with valence 3,5,6, whereas weakly balanced grids also contain valence 7 singular edges, which appear when 4 out of the 5 additional schemes are used. Since our schemes can be used as-is, without further modification, no edges with valence different from the ones declared here are possible.

Considering the portion of Thingi10K where [Gao et al. 2019] produced a valid output, our method resulted superior to its competitor in that it never introduced singular edges with valence higher than 6, whereas the schemes of Gao and colleagues introduced valence 7

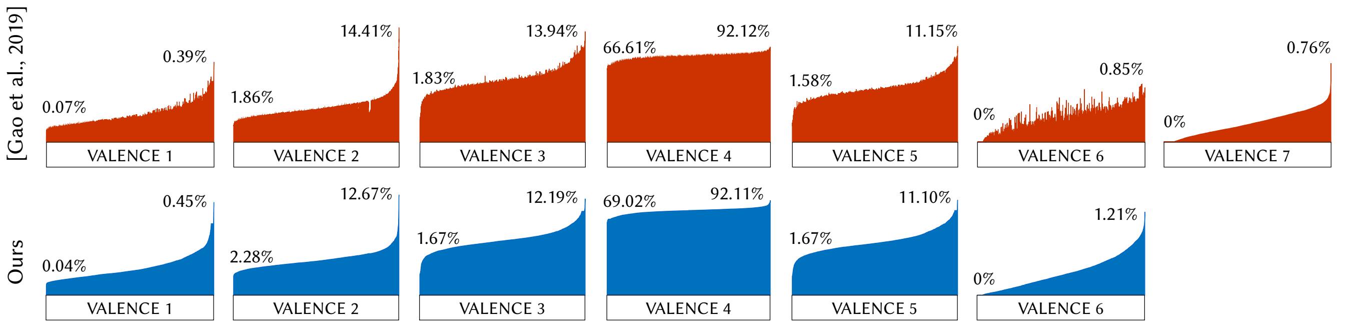

Fig. 18. Distribution of edge valences for 7589 models from Thingi10K. Each column corresponds to a specific model in the dataset and its height is proportional to the relative impact of that valence in the output mesh. Lateral numbers indicate the minimum and maximum relative impacts. The 37 failure cases of [Gao et al. 2019] were omitted from the analysis.

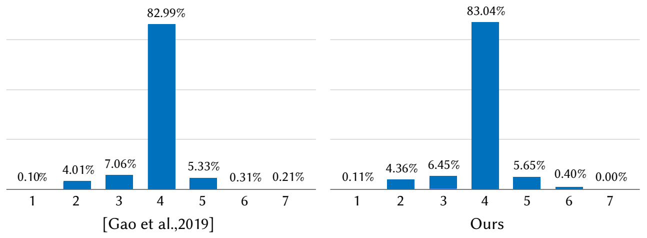

Fig. 19. Cumulative relative impact of edge valences across all models in the Thingi10K dataset. The vast majority of edges is regular (valence 4), and the impact of other valences is very similar for both methods, with the exception of valence 7 edges, which are completely avoided by our schemes if the input grid is strongly balanced.

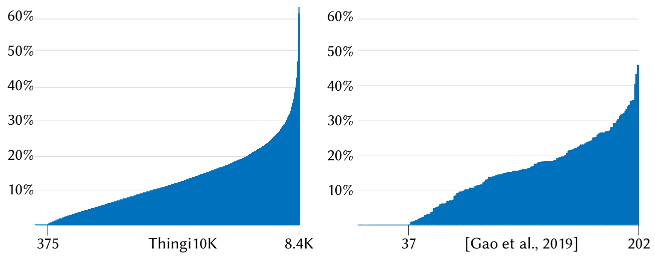

Fig. 20. Strongly balanced grids required up to 64% more elements than weakly balanced grids in the Thingi10K dataset, and up to 46% more elements in the dataset released with [Gao et al. 2019]. In the plots above, each column represents a different shape in the dataset, and columns are ordered for increasing growth. Mesh growth was measured as $(H_S - H_W)/H_W$, with H_S and H_W being the number of grid cells obtained applying the strong and weak balancing, respectively.

edges in 7072 cases out of 7589 (93.2%). In Figure 19 we show a comparative analysis of the relative distribution of edge valences. As can be noticed, the complete absence of valence 7 edges in our output meshes does not impact the distributions of alternative valences, which appear almost identical to the ones of our competitor. A more detailed overview of valence distribution is depicted in Figure 18.

Avoiding high valence edges provides twofold advantages. On the one hand, irregular edges with high valence impose tighter bounds on maximum per element quality (Figure 2). Furthermore, they are more difficult to handle for modern local/global untangling methods such as [Aigerman and Lipman 2013; Livesu et al. 2015; Marschner et al. 2020; Overby et al. 2021]. In fact, these tools operate by first computing a locally optimal solution for each element separately, and then reconcile all local solutions in a global step. This approach intrinsically suffers the presence of high valence mesh elements, because the number of alternative local solutions to be combined grows, making it harder to find consensus between all of them.

Mesh size. In the general case, an adaptive grid that has been split to faithfully approximate a target geometry cannot be readily transformed into a conforming hexahedral mesh. For this to be possible, the grid must undergo two processing steps: (i) additional refinement must be applied in order to fulfill minimal topological requirements for the application of the transition schemes; (ii) grid elements around the hanging nodes must be substituted with small clusters of transition elements. We empirically observed that the first operation consistently impacts mesh size much more than the latter. This happened for all the test models in Thingi10k, where step (i) more than doubled the original grid size on average and increased it by a factor of 9x in the worst case, whereas the impact of step (ii) was around 20% of the original grid size on average. Being able to operate on a wider class of adaptive grids, our transition schemes mitigate the impact of step (i) on mesh size. This is because processing a generic grid to fulfill weak balancing is likely to require less refinement than the one necessary to fulfill the strong balancing criterion required by the schemes proposed in [Gao et al. 2019; Maréchal 2009]. To give some numbers, for the dataset released with [Gao et al. 2019], in 167 out of 202 cases (82.7%) weakly balanced grids were coarser than strongly balanced ones. For Thingi10K this happened on more than 8K cases out of 8.4K (95.6%). The extent of the size reduction largely depends on the refinement patterns induced by the input geometry, which in turn depends on the shape morphology and the octree splitting rule used. In our experiments, strongly balanced grids required 16% more elements than weakly balanced grids on average, and 64% more elements in the worst case. The histograms in Figure 20 show the details of this comparison for all the tested models. Considering that most applications involving hexmeshes require solving a global linear system, and that the computational cost of a linear solve scales cubically with the number of mesh vertices [Krishnamoorthy and Menon 2013], weak balancing promises to introduce a significant speedup for applications. Perhaps a more concrete evidence comes from a recent technical report published by Ferrari, where the car maker declares that by reducing cell count by 15% it was able to run 300% more CFD simulations, helping the engineers to develop their cars or new solutions faster [Ferrari 2020] .

8 CONCLUSIONS

We have extensively studied the topological schemes that permit to transform an adaptively refined grid into a pure hexahedral mesh. Previous literature had already proved that directly incorporating hanging nodes into the hexahedral mesh is not always possible. Therefore, our analysis and contributions are restricted to dual schemes, which aim to generate a general polyhedral mesh that yields only hexahedral elements when dualized.

We started our study from the seminal work of Maréchal [2009], who pioneered dual approaches. We have shown that both his schemes and the schemes proposed in later articles are not exhaustive and do not contemplate ambiguities that arise when one tries to implement them. We explicitly describe and release, for the first time, a set of schemes that fully cover the combinatorial space of adaptive balanced grids, also relaxing the notion of balancing from strict – as it was used in prior art – to weak. As a result, we were able to enlarge the class of grids that can be transformed into conforming hexahedral meshes, showing with an extensive empirical analysis that the meshes produced with our method are significantly superior than prior art in terms of ability to produce a valid result (i.e., conforming, all-hex), singular structure and element count.

At this stage, we believe that major improvements are unlikely to come from alternative schemes for adaptive grids that are already supported by the current ones, but rather on novel ideas to embrace a broader class of input grids. In fact, based on our analysis the current bottleneck in the pipeline is the amount of refinement that adaptive grids must undergo to ensure the applicability of the transitions. Our extension to weakly balanced grids is a first step in this direction, because it opens hexmeshing to a new class of inputs. We believe that more can be done in this regard, and we will devote our future efforts to working in this direction.

Limitations and future works. While the transition schemes proposed in this paper are *topologically optimal*, in the sense that they minimize the extent of high valence irregular edges, they do not guarantee that an embedded mesh with this connectivity will be superior to a mesh obtained with alternative methods. Finding the embedding that maximizes the geometric quality of a certain mesh (e.g. to evaluate the quality of its connectivity) is an complex problem for which there exists no solution to our knowledge. The best smoothing and untangling methods of which we are aware aim to squeeze the maximum potential from a certain connectivity with heuristic approaches. As mentioned in Section 7, since most of these methods are local/global, we conjecture that they should benefit from our findings, and we will devote part of our future works to further investigate this topic.

ACKNOWLEDGEMENTS

We are deeply grateful to the authors of [Gao et al. 2019] for the many clarifications on their article and source code, and for releasing a dataset of meshes produced both with their tool and with [Maréchal 2009]. We also thank the Hexalab project [Bracci et al. 2019], which we extensively used during the creation of this article. The work of Marco Livesu was partly supported by EU ERC Advanced Grant CHANGE No. 694515. Gianmarco Cherchi gratefully acknowledges the support to his research by PON R&I 2014-2020 AIM1895943-1.

REFERENCES

Noam Aigerman and Yaron Lipman. 2013. Injective and bounded distortion mappings in 3D. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–14.

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.

Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Dual loops meshing: quality quad layouts on manifolds. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Lih-Shyang Chen, Gabor T Herman, R Anthony Reynolds, and Jayaram K Udupa. 1985. Surface shading in the cuberille environment. *IEEE Computer Graphics and Applications* 5, 12 (1985), 33–43.

Gianmarco Cherchi, Pierre Alliez, Riccardo Scateni, Max Lyon, and David Bommes. 2019. Selective padding for polycube-based hexahedral meshing. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 580–591.

CoreForm. 2020. Cubit. https://coreform.com/products/coreform-cubit/

Etienne Corman and Keenan Crane. 2019. Symmetric Moving Frames. *ACM Trans. Graph.* 38, 4 (2019).

Dassault Systèmes. 2020. Spatial Corp. https://www.spatial.com/products/3d-precise-mesh

Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–9.

Ferrari. 2020. Tech Insight: CFD Design in the 488GTE. https://www.ferrari.com/en-EN/competizioni-gt/articles/tech-insight-cfd-design-in-the-488-gte

Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient volumetric polycube-map construction. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 97–106.

Xifeng Gao, Tobias Martin, Sai Deng, Elaine Cohen, Zhigang Deng, and Guoning Chen. 2015. Structured volume decomposition via generalized sweeping. *IEEE transactions on visualization and computer graphics* 22, 7 (2015), 1899–1911.

Xifeng Gao, Hanxiao Shen, and Daniele Panozzo. 2019. Feature Preserving Octree-Based Hexahedral Meshing. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 135–149.

James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-hex mesh generation via volumetric polycube deformation. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 1407–1416.

Gabor T Herman and Hsun Kao Liu. 1979. Three-dimensional display of human organs from computed tomograms. *Computer graphics and image processing* 9, 1 (1979), 1–21.

Kangkang Hu, Jin Qian, and Yongjie Zhang. 2013. Adaptive all-hexahedral mesh generation based on a hybrid octree and bubble packing. *22nd International Meshing Roundtable* (2013).

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. ℓ1-based construction of polycube maps from complex shapes. *ACM Transactions on Graphics (TOG)* 33, 3 (2014), 1–11.

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.

Michael Kremer, David Bommes, Isaak Lim, and Leif Kobbelt. 2014. Advanced automatic hexahedral mesh generation from surface quad meshes. In *Proceedings of the 22nd International Meshing Roundtable*. Springer, 147–164.

Aravindh Krishnamoorthy and Deepak Menon. 2013. Matrix inversion using Cholesky decomposition. In *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*. IEEE, 70–72.

Bruno Lévy and Yang Liu. 2010. L p centroidal voronoi tessellation and its applications. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–11.

Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. 2012. All-hex meshing using singularity-restricted field. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.

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 (2015), 107–117.

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), 93–1.

Marco Livesu. 2019. cinolib: a generic programming header only C++ library for processing polygonal and polyhedral meshes. *Transactions on Computational Science XXXIV* (2019). https://doi.org/10.1007/978-3-662-59958-7_4 https://github.com/mlivesu/cinolib/.

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. https://doi.org/10.1111/cgf.13021

Marco Livesu, Nico Pietroni, Enrico Puppo, Alla Sheffer, and Paolo Cignoni. 2020. LoopyCuts: Practical Feature-Preserving Block Decomposition for Strongly Hex-Dominant Meshing. *ACM Transactions on Graphics (SIGGRAPH)* 39, 4 (2020). https://doi.org/(10.1145/3386569.3392472)

Marco Livesu, Alla Sheffer, Nicholas Vining, and Marco Tarini. 2015. Practical hex-mesh optimization via edge-cone rectification. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. 2013. PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA 2013)* 32, 6 (2013). https://doi.org/10.1145/2508363.2508388

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* 21, 4 (1987), 163–169.

Loïc Maréchal. 2009. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *Proceedings of the 18th international meshing roundtable*. Springer, 65–84.

Loïc Maréchal. 2016. All hexahedral boundary layers generation. *Procedia engineering* 163 (2016), 5–19.

Zoë Marschner, David Palmer, Paul Zhang, and Justin Solomon. 2020. Hexahedral Mesh Repair via Sum-of-Squares Relaxation. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 133–147.

Scott A Mitchell. 1996. A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volume. In *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 465–476.

Gregory M Nielson. 2004. Dual marching cubes. In *IEEE Visualization 2004*. IEEE, 489–496.

Matthias Nieser, Ulrich Reitebuch, and Konrad Polthier. 2011. Cubecover–parameterization of 3d volumes. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 1397–1406.

Matthew Overby, Danny Kaufman, and Rahul Narain. 2021. Globally Injective Geometry Optimization with Non-Injective Steps. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.14361

Teseo Schneider, Jérémie Dumas, Xifeng Gao, Mario Botsch, Daniele Panozzo, and Denis Zorin. 2019. Poly-spline finite-element method. *ACM Transactions on Graphics (TOG)* 38, 3 (2019), 1–16.

Robert Schneiders. 1996. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with computers* 12, 3-4 (1996), 168–177.

Robert Schneiders. 1997. An algorithm for the generation of hexahedral element meshes based on an octree technique. In *6th International Meshing Roundtable*. 195–196.

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

Robert Schneiders. 2000b. Octree-based hexahedral mesh generation. *International Journal of Computational Geometry & Applications* 10, 04 (2000), 383–398.

Robert Schneiders, Roland Schindler, and Frank Weiler. 1996. Octree-based generation of hexahedral element meshes. In *IN PROCEEDINGS OF THE 5TH INTERNATIONAL MESHING ROUNDTABLE*. Citeseer.

Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer* 24, 4 (2008), 249–259.

Mark S Shephard and Marcel K Georges. 1991. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical methods in engineering* 32, 4 (1991), 709–749.

Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary element octahedral fields in volumes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.

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.

Timothy J Tautges and Sarah E Knoop. 2003. Topology modification of hexahedral meshes using atomic dual-based operations. *algorithms* 11 (2003), 12.

Erke Wang, Thomas Nelson, and Rainer Rauch. 2004. Back to elements-tetrahedra vs. hexahedra. In *Proceedings of the 2004 international ANSYS conference*. Ansys Pennsylvania.

Wei Wang, Yong Cao, and Tsubasa Okaze. 2021. Comparison of hexahedral, tetrahedral and polyhedral cells for reproducing the wind field around an isolated building by LES. *Building and Environment* (2021), 107717.

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).

Yongjie Zhang and Chandrajit Bajaj. 2006. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer methods in applied mechanics and engineering* 195, 9-12 (2006), 942–960.

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