DOI: 10.1111/cgf.14470
EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)
Volume 41 (2022), Number 2

The 3D Motorcycle Complex for Structured Volume Decomposition

Hendrik Brückler Ojaswi Gupta Manish Mandad Marcel Campen

Osnabrück University, Germany

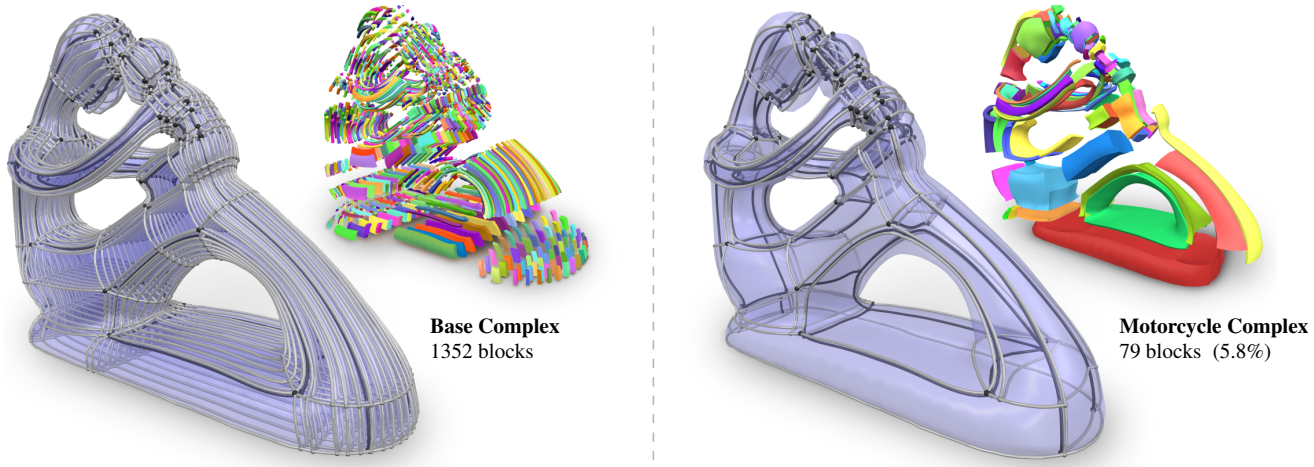


Figure 1: *Base Complex (left) and our Motorcycle Complex (right) induced by the same volumetric seamless parametrization of a solid object, both providing a structured partition into cuboid blocks. The motorcycle complex often partitions the object's interior into a much smaller number of blocks, here just 5.8%, 79 instead of 1352 blocks (see the exploded views). We provide a definition of this motorcycle complex, describe algorithms for its construction, and demonstrate its use and benefits.*

Abstract

The so-called motorcycle graph has been employed in recent years for various purposes in the context of structured and aligned block decomposition of 2D shapes and 2-manifold surfaces. Applications are in the fields of surface parametrization, spline space construction, semi-structured quad mesh generation, or geometry data compression. We describe a generalization of this motorcycle graph concept to the three-dimensional volumetric setting. Through careful extensions aware of topological intricacies of this higher-dimensional setting, we are able to guarantee important block decomposition properties also in this case. We describe algorithms for the construction of this 3D motorcycle complex on the basis of either hexahedral meshes or seamless volumetric parametrizations. Its utility is illustrated on examples in hexahedral mesh generation and volumetric T-spline construction.

Keywords: block-structured, multi-block, T-mesh, hexahedral mesh, volume mesh, block decomposition, base complex

CCS Concepts

• **Computing methodologies** → **Computer graphics; Mesh models; Mesh geometry models;** *Shape modeling;*

1. Introduction

The motorcycle graph [EGKT08, EE99] has been used in various computer graphics and geometry processing applications to partition surfaces in a structured manner, as discussed further in Sec. 2. Conceptually, a number of particles (called *motorcycles*) are traced over a surface, each one stopping when reaching a trace. The collection of traces finally forms a surface-embedded graph that partitions the surface. This idea has been used on surfaces equipped with various structures that define the directions the motorcycles take, most relevantly:

- cross fields or frame fields,
- seamless or integer-grid parametrizations,
- quadrilateral meshes.

These objects all impose a structure on the surface that defines four directions everywhere, except at a number of isolated singularities. Under mild assumptions, the motorcycle graph, with particles

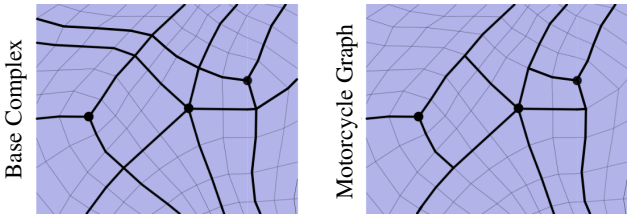


Figure 2: *Left: base complex (black) of a quad mesh (grey edges), providing a conforming partition. Right: motorcycle graph, providing a non-conforming partition, with 4 T-joints in this example.*

starting at these singularities, yields a partition of the surface into patches that all are *four-sided*, completely *regular* in their interior, and *aligned* with the field's streamlines, the parametrization's isolines, or the mesh's edges, respectively.

A related structure is the so-called base complex [BLK11], also referred to as quad layout [CK14, PPM*16]. It can be obtained by not letting particles stop at traces (but only at singularities or boundaries). The base complex is known to be the coarsest *conforming* partition into four-sided, regular, aligned patches. By contrast, the partition obtained by the motorcycle graph is typically *non-conforming*: there can be T-joints, as illustrated in Fig. 2.

For use cases that are able to handle this non-conformity (or even benefit from it), the motorcycle graph provides a major advantage: it is often much simpler (having a smaller number of patches), in some cases even by orders of magnitude, than the base complex. It has been exploited in recent years (see Sec. 2) for scenarios like

- generation of quad meshes [MPZ14],
- localized structured remeshing [NHE*19],
- quantization of global parametrizations [CBK15],
- construction of T-spline spaces [CZ17],
- texture mapping of surfaces [SPGT18],
- mesh-based computational fabrication [LLZ*20].

1.1. Contribution

We propose the *motorcycle complex*, a generalization of the 2D motorcycle graph to the 3D volumetric setting. In analogy to the 2D case, it partitions a volumetric object, equipped with a suitable directional structure, into regular cuboid blocks (rather than quadrilateral patches) in a non-conforming manner, cf. Fig. 1. Suitable guiding structures are volumetric seamless parametrizations, 3D integer-grid maps, and hexahedral meshes.

Algorithmic tools necessary to compute this motorcycle complex, based either on parametrized tetrahedral meshes or on hexahedral meshes, are introduced. We analyze the characteristics of the resulting complex, and show that important properties are guaranteed by the induced partition. Most importantly, this includes the cuboidal structure and the regularity of each induced cell.

Looking at the ways the motorcycle graph has been successfully leveraged in the 2D case (in particular when it comes to guaranteeing robustness), this motorcycle complex has the potential to serve as foundation for important advances in hexahedral mesh generation, volumetric T-spline definition, and further problems related to grid generation, volumetric parametrization, and isogeometric analysis. We illustrate this with two example applications in Sec. 7.

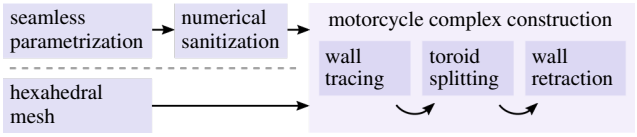


Figure 3: *We take as input either a volumetric seamless parametrization (on a tetrahedral mesh) or a hexahedral mesh, and compute an induced motorcycle complex in three algorithmic steps.*

1.2. Overview

Input Following Fig. 3, the input to our method is a seamless parametrization (Sec. 3.1) on a tetrahedral mesh, sanitized numerically (Sec. 6) for robustness if necessary. Alternatively a (non-parametrized) hexahedral mesh can be considered—analogously to how the motorcycle graph in 2D has been used on suitably parametrized triangle meshes as well as on quadrilateral meshes. While the case of seamless parametrization input is most relevant (and algorithmically more challenging and interesting), we discuss the simpler hexahedral mesh case as a more intuitive entry, too.

Goal The input object is to be partitioned into a small number of cuboid blocks (see Fig. 1) such that they are regular in their interior (not containing any singularities or irregular vertices/edges, respectively) and each of a block's six boundary patches is aligned with an iso-surface in the parametrization or a sheet of quads in the hexahedral mesh, respectively. In other words, the blocks are axis-aligned rectangular cuboids under the parametrization, or regular $l \times m \times n$-grid pieces of the hexahedral mesh, respectively.

Approach The goal is met by constructing a motorcycle complex induced by the input, as defined in Sec. 4. This is done in three algorithmic steps (see Fig. 3 right), detailed in Sec. 5. In step 1, parts of the iso-surfaces incident at the singularities are incrementally designated as block walls in an expansion process. In step 2, possibly additional walls are designated to guarantee the desired block decomposition property. In step 3, redundant walls are retracted, in regions where the initial expansion process was over-zealous.

2. Related Work

The original 2D Euclidean definition of the motorcycle graph goes back to work by Eppstein and Erickson [EE99]. It was extended to curved surfaces, i.e., Riemannian manifolds, initially for the purpose of quadrilateral mesh partitioning [EGKT08]. In this setting the mesh's edges provide the directional information guiding the motorcycles across the surfaces. This quad mesh driven motorcycle graph has been employed in further contexts, for reverse engineering [GMSO14], texture mapping [SPGT18], computational fabrication [LLZ*20], and quadrilateral remeshing [NHE*19, RP17].

The idea of the motorcycle graph has been adapted to surfaces equipped with other directional guiding structures. In particular, motorcycles following streamlines of a cross field [VCD*16] have been used for the reliable generation of global seamless surface parametrizations [MPZ14]. Motorcycles following the isolines of such seamless parametrizations [KNP07, BZK09, MZ12], in turn, have been used for the purpose of robust parametrization quantization [CBK15, LCBK19, LCK21a, LCK21b]. This re-

sults in integer-grid maps, which are important ingredients for the generation of quadrilateral meshes [BCE*13]. A generalized class of parametrizations, so-called seamless similarity parametrizations, provide another structure that can be used to guide motorcycles [CZ17]. This has been leveraged for the reliable construction of T-meshes that can serve as domain for the definition of T-spline spaces [CZ17, KPP17].

For the 3D volumetric case, a concept analogous to the 2D motorcycle graph has not been described yet. Generalizations of the above mentioned guiding structures, however, often do exist. Hexahedral meshes can be considered the natural generalization of quadrilateral meshes to the next dimension. Cross fields generalize to octahedral fields [SVB17, HTWB11, LZC*18, CC19, ZVC*20], and seamless parametrizations of triangular surface meshes extend naturally to tetrahedral volume meshes as well [NRP11], see also Sec. 3.1.

So far only the base complex [BLK11]§2.2 (which also provides an aligned decomposition into regular blocks) has been considered in a 3D setting, for the case of hexahedral meshes [GDC15]. For hexahedral meshes with many details, this structure can be highly complex; even more so for seamless volume parametrizations (which can be viewed as infinitely dense hexahedral meshes), where it can easily become impractically large.

More distantly related are volumetric block decomposition algorithms not driven by a prescribed singularity structure or targeting other use cases, based on plastering [SKO*10], medial axes [SERB99], or cut sheets [Tak19, LPP*20].

3. Background

3.1. Seamless Parametrization

Given a surface M, a *seamless (surface) parametrization* [MZ12] is a chart-based map $\phi : M^c \to \mathbb{R}^2$ (where M^c is M cut to one or more topological disks) such that chart transitions are rigid, with a rotation by some multiple of $\pi/2$. Analogously, given a volume M, a *seamless (volume) parametrization* is a chart-based map $\phi : M^c \to \mathbb{R}^3$ (where M^c is M cut to one or more topological balls) such that chart transitions are rigid, with a rotation from the octahedral rotation group [NRP11].

In the discrete 3D case, with M given as a tetrahedral mesh, we assume ϕ to be affine per tetrahedron, with transitions across facets. Unless stated otherwise, a seamless parametrization is assumed to be *valid* (non-degenerate and orientation preserving) and such that boundary facets of M are aligned. A facet (edge) is said to be *aligned* if its image under ϕ is constant in one (two) coordinate components, i.e., it is parallel to one coordinate plane (axis). The sum of incident parametric dihedral angles around an edge of M is a multiple of $\pi/2$; an edge is *regular* if it is 2π for interior, or π for boundary edges; otherwise it is *singular*. Like all known use cases we require singular edges to be aligned. For the practically by far most relevant types of singularities, deviating from the regular case by $\pm\pi/2$ [LZC*18], this is inherent anyway [EBCK13]; for others, constraints can ensure it [NRP11].

An *integer-grid parametrization* [BCE*13] (also *quantized* parametrization [CBK15]) is a special case (Fig. 4): the translational components of all transitions as well as the constant components of all images of singular and aligned elements are from $\mathbb{Z}$ instead of $\mathbb{R}$. Such an integer-grid parametrization (in the 3D case) naturally induces a hexahedral mesh [NRP11]. Singularities induce irregularities in the mesh (edges with more or less than 4 adjacent hexahedra), and sheets of quads in the mesh coincide with iso-surfaces of the parametrization.

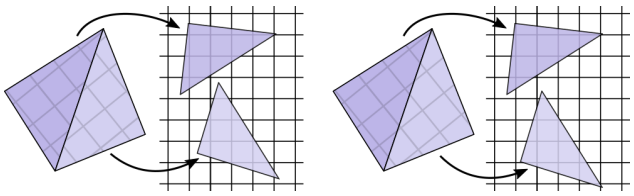


Figure 4: *Illustration of a 2D seamless parametrization. Left:* continuous. *Right:* quantized. *In both cases, there is no scale discontinuity across the chart transition between the two triangles, and the rotation is some multiple of $\pi/2$. On the right, additionally, the translation is integral, making the integer grid continuous.*

We do not make any implicit assumption about parametrizations being quantized in the following. Where explicit distinction is necessary, we use the terms *continuous* parametrization versus *quantized* parametrization. Importantly, our method is able to operate on arbitrary valid continuous seamless parametrizations. As discussed in Sec. 7 the motorcycle complex can actually be used to yield a quantized parametrization (and therefore also a hex mesh) from a continuous one—a task hard to solve with previous techniques.

Metric We will argue about curves or surfaces in M being straight, planar, or orthogonal *with respect to* ϕ or *in the* ϕ*-metric*. This is to be understood as measuring these objects' images under ϕ in $\mathbb{R}^3$—or equivalently: measuring in M using the metric tensor that is the pull-back through ϕ of the Euclidean metric tensor.

3.2. 2D Motorcycle Graph

Various incarnations of the 2D motorcycle graph idea have been used on surfaces. For the case of a seamless parametrization ϕ providing guidance on surface M, it can be summarized as follows. At each point $p_i \in M$ where ϕ is singular, for each direction d_i of an incident iso-line of ϕ a particle (p_i, d_i) is placed. Simultaneously, each particle starts tracing (with unit speed, from p_i in direction d_i) a curve across M that is straight with respect to ϕ, i.e., it is an iso-curve (taking transitions into account). A particle stops when it hits: (i) a trace (left behind by itself or another particle), (ii) a point where ϕ is singular, or (iii) the boundary of M. Upon termination, the collection of traces forms a surface-embedded graph, the motorcycle graph. When instead ignoring stopping criterion (i), the resulting graph is the base complex (see Fig. 2). Clearly, the motorcycle graph is a subgraph of the base complex graph.

The following properties were shown for the (non-empty) motorcycle graph [EGKT08]:

- each patch has disk topology,
- each patch has four sides, aligned with isolines,
- each patch is regular, i.e., free of interior singularities.

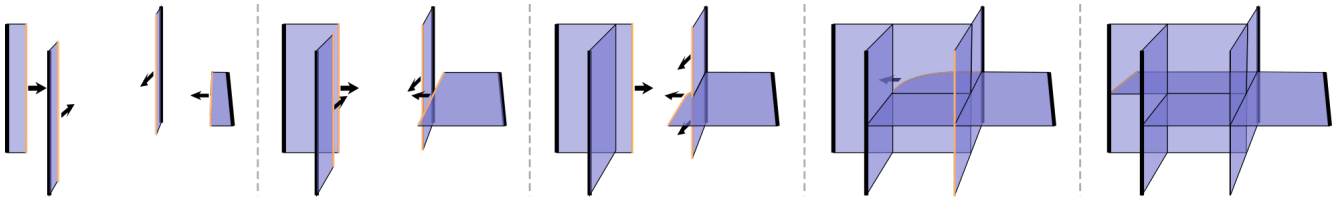


Figure 5: *Illustration of the brush fire process in 3D. For simplicity and visual clarity only a single iso-surface per singular curve (bold black) is shown, clipped to a cubical region, in a setting where iso-surfaces are planar. The fire front is highlighted in orange, its conceptual direction of expansion is indicated by arrows. The supplementary video gives an animated impression of the process in more complex settings.*

Furthermore, the number of patches is within a constant factor of the minimum number possible for any partition with these properties. Finding a truly minimal partition is known to be much harder than computing the motorcycle graph [EGKT08].

4. The Motorcycle Complex

The idea behind the motorcycle graph does not generalize easily to higher dimensions. While in 2D *curves* (traces of particles) are sufficient to partition the two-manifold into patches, in 3D *surfaces* are required to partition the manifold into blocks. These cannot be modeled as traces of some finite number of moving point particles. Instead, we interpret the construction process as an equivalent brush fire expansion process, in such a way that it is dimension-generic. Conceptually, a fire is ignited simultaneously at all points on singularities of ϕ. It is confined to spread within $(n-1)$-dimensional isoparametric submanifolds that contain the singularities, and cannot cross points already burnt. If M has a boundary, it is additionally considered burnt.

For $n = 2$ the singularities are points and the 1-dimensional isoparametric submanifolds are iso-curves of ϕ, i.e., curves $c(t)$ along which $\phi(c(t)) = (u,v)$ is constant in either u or v (taking chart transitions into account). This coincides with the classical definition of the 2D motorcycle graph.

For $n = 3$ the singularities are curves [LZC*18] and the 2-dimensional isoparametric submanifolds are surfaces $c(s,t)$ on which $\phi(c(s,t)) = (u,v,w)$ is constant in either u, v, or w. Note that all singular curves are isoparametric curves (Sec. 3.1), i.e., they are contained in such isoparametric surfaces, such that the fire starting on different points of a singular curve will spread in common iso-surfaces. The example in Fig. 5 illustrates the concept. We discuss the properties of the implied decomposition of M in Sec. 4.1.

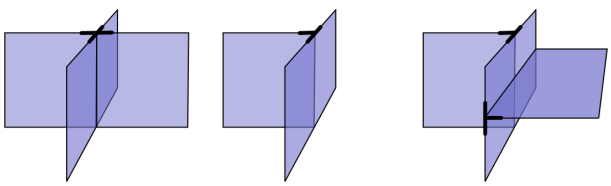


Figure 6: *Left: regular crossing, formed by four walls. Center: T-joint, formed by three walls. Right: interaction of two T-joints; in contrast to the other two configurations (essentially extrusions of their 2D counterparts) this is a configuration specific to the 3D case.*

Let us point out an important difference between the 2D and the 3D case: In 2D, the fire front at any time consists of a set of isolated points. Whenever such a point reaches a location already burnt, it dies. This gave rise to the original motorcycle metaphor. In 3D, the fire front is a set of curves (a continuum of points). Such a curve may partially reach burnt terrain, and the remainder proceeds (flowing around the obstacle; Fig. 5 center). The motorcycle metaphor thus, in contrast to the confined brush fire, applies only loosely to the *process* in 3D, but we adopt the name due to the very close analogy in terms of its *results*, the partitions and their properties. Note that considering the fire front curves as atomic entities instead, that completely stop when any part reaches burnt terrain, would not yield the desired partition properties discussed in the following.

4.1. Properties

We define the following terminology:

- *node*: intersection point of multiple (non-colinear) arcs.
- *arc*: intersection curve of multiple (non-coplanar) walls.
- *wall*: part of the burnt space bounded by arcs.
- *block*: part of M bounded by walls.

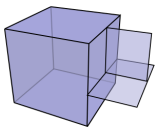
These entities form the 0-, 1-, 2-, and 3-dimensional cells of a non-conforming generalized cell complex; in contrast to the usual definition of a cell complex, the k-cells implied by the brush fire construction are not necessarily homeomorphic to a k-ball. We discuss (and resolve) this in the following.

4.1.1. Block Regularity

By construction, all singular points of ϕ are contained in arcs or nodes. The interior of each block (as well as the interior of each wall) therefore contains only regular points; when restricted to a single block b, $\phi|_b$ is regular.

4.1.2. Element Types

First, we can observe that blocks have a boundary that is piecewise planar w.r.t. ϕ; we call each planar piece a *block facet*. This is because a block is bounded by walls, and walls are isoplanes of ϕ. Note that a block facet may consist of one or of multiple walls; in the inset the right block facet consists of three walls, due to T-joints implied by external walls incident at the block. We will establish that the facets of a block meet only in 90°edges (at arcs) and in "90°corners" (i.e., solid corners where three 90°edges meet). These angles are to be understood w.r.t. ϕ.



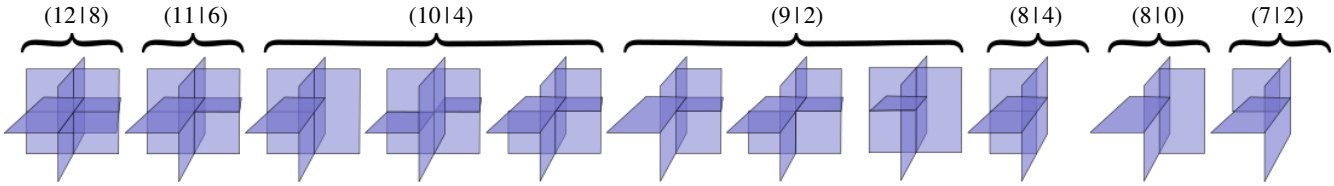
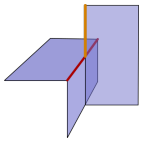


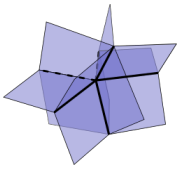
Figure 7: *All wall configurations that may occur around a single node in a regular region (up to symmetry). The values (w|c) specify the numbers of walls w and the numbers of solid corners c (all of* 90°*type) incident at the node. Any other configuration cannot occur in the motorcycle complex because it contains open edges or* 270°*edges (see Sec. 4.1.2).*

Edges Around singular curves, isoparametric surfaces emanate in 90°intervals. Therefore, at singularities, blocks have 90°edges. Away from singularities, walls end only where they hit another wall or the model's boundary. In both cases, because both walls and the boundary are (piecewise) iso-parametric (and different iso-planes are orthogonal in ϕ), 90°edges are formed (cf. Fig. 6).

Regular Corners Solid corners are formed wherever more than two walls meet in one point. This can be at singularities (where walls emanate in the brush fire process) and at points away from singularities where multiple planes meet in the course of the brush fire process. Fig. 7 lists all the possible wall configurations that may occur at such a regular point. Obviously, they are all subsets of the *complete* configuration labeled (12|8), with the maximum of 12 walls meeting in one point. All other subsets (those not depicted) contain an open edge or a 270°edge, as illustrated in the inset in orange and red. Open edges cannot occur as the brush fire would not have stopped there; 270°edges cannot occur because at least one of the two incident walls would have continued.



Singular Corners At a singular point the configuration looks different, and depends on the singularity type (which there are infinitely many of [LZC*18]). In any case, however, if a corner would be formed that is not a 90°corner, this would imply there is an incident block edge that is not a 90°edge. At singular curves (bold black in the inset), however, only 90°edges are formed, and around potential additional regular isolines incident to singular points (dotted) the situation is analogous to the above regular case: open edges and 270°edges cannot occur, so only 90°edges are possible.



Facet Types Given this restriction to 90°corners, following the Gauss-Bonnet theorem a block's facet (consisting of one or more walls) must be a disk with 4 corners (a rectangle), or an annulus with no corners. Closed (toroidal) facets, without any incident singularity or wall, could only occur (on the boundary) in the trivial case of an entirely regular ϕ. A rectangle facet cannot be adjacent to an annulus facet of the same block: at the corners of a rectangle, glued to an annulus at a 90°edge, either a corner in the annulus, or an adjacent 180°edge would be implied, both of which we ruled out. A block thus has either rectangular or annulus facets, not both.

Block Types For a block the Gauss-Bonnet theorem, together with the restriction to 90°corners, implies that its surface must be either of genus 0 with 8 such corners, or of genus 1 with no corners.

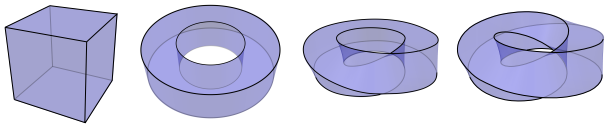
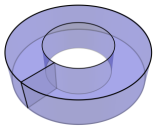


Figure 8: *Blocks of the raw motorcycle complex can only be cuboidal or toroidal. Toroidal blocks may have 4, 1, or 2 annulus facets depending on their* twist *(here 0,* $\frac{1}{4}$*, and* $\frac{1}{2}$*, respectively).*

A block of genus 0 cannot have annulus facets, as they would not form any corners. According to Euler's formula, it must therefore have 6 (rectangular) facets. There are only two (structurally distinct) polyhedra with 8 vertices and 6 faces: the cube and the tetragonal antiwedge. Of these only the cube has four-sided facets. We conclude that if a block is simply-connected, it must be a cuboid (in particular a rectangular cuboid with respect to the ϕ-metric).

For a block of genus 1, facets must be annuli, so as to not form any corner. Blocks then must be tori, with rectangular cross section. There is an infinite number of structurally different such tori: the *twist* of the torus can be an arbitrary number of quarter turns. In case of twist $k\frac{1}{4}$, the block has 4 facets if $k \mod 4 = 0$, 2 facets if $k \mod 4 = 2$, and only 1 facet if k is odd. Fig. 8 illustrates these types of blocks that initially can occur in the motorcycle complex.

Any genus 1 block, regardless of its twist, can be turned into a (self-adjacent) genus 0 block by introducing one additional wall that cuts it. Using this modification (Sec. 5.3), a pure cuboid block complex is obtained in any case, avoiding the need for further special case handling in subsequent operations.



4.2. Reducibility

Just as in the 2D case (Sec. 3.2), we cannot expect the resulting motorcycle complex to describe a globally minimal (i.e., smallest) partition with the desired properties (cuboidal, regular, aligned). It is worthwhile considering the aspect of local minimality, though. To this end we define the following (for the 3D and 2D cases):

Definition 1 (Reduction)**.** *The operation of merging two adjacent n-cells of a cuboid (quadrilateral) cell complex into one n-cell that is cuboid (quadrilateral) we call a* reduction*. A complex that allows for no such reduction of any two n-cells we call* irreducible*.*

Two n-cells are adjacent if they share an $(n-1)$-cell (an arc in the 2D case, a wall in the 3D case). If they can be merged in a reduction, we say the shared cell is *removable*.

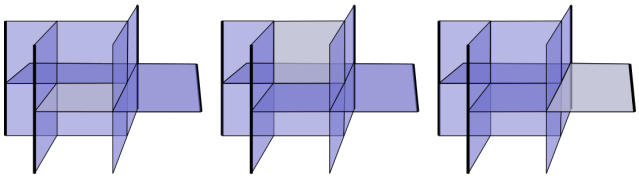


Figure 9: *Illustration of* reducibility*, based on the example from Fig. 5. Left: the grey wall can be removed, merging two cuboid cells into one that is still cuboid. Center: the grey wall cannot be removed as it would yield a non-cuboid block. Right: the grey wall is not regular-removable as it is incident to a singular arc (that would end up lying inside the merged block's facet).*

A restricted notion is that of *regular-irreducibility*, defined via *regular-reductions* that merge cells only across *regular-removable* arcs or walls not incident to any singular node or arc. This notion is relevant for use cases (such as in Sec. 7.2) that require singularities lie only at block edges, not in the interior of block facets.

Under a general position assumption on the location of singularities, the standard 2D motorcycle graph (while reducible) is regular-irreducible; only when motorcycles meet in a frontal manner there may be options for regular-reduction. As demonstrated in Fig. 9, the situation is different in the 3D case: even regular-irreducibility is not a given, the brush fire result commonly contains regular-removable walls. The underlying reason is related to the discussion at the beginning of Sec. 4: while motorcycles in 2D are points, and collisions with traces are isolated instantaneous events, in 3D the more complex brush fire that forms a wall may stop in one place while proceeding in another.

Wall Retraction We therefore propose to subsequently reduce the result of the brush fire process to a locally minimal, i.e., either regular-irreducible or irreducible state, as desired. To this end, we perform *wall retraction*: (regular-)removable walls are greedily removed, ordered by their parametric distance from their origin. Intuitively, this can be interpreted as retracting fire walls in places where they have spread unnecessarily far in the brush fire expansion. It would be conceptually attractive to avoid this redundancy already during the expansion process, but this is not straightforward. Practically, the overhead due to the reduction happening after the fact is benign (see experiments in Sec. 7).

Where distinction is necessary, we refer to the non-reduced brush fire result as *raw motorcycle complex*, while *motorcycle complex* is generally meant to refer to the reduced version (with additional walls inserted in rare toroidal cells, Sec. 4.1.2). In Sec. 5 we describe the construction as well as the reduction process in detail.

Remark (Base Complex Reduction) One could start from the base complex and apply reductions until an irreducible minimum is achieved. However, the base complex can be very large, hampering practical construction, and, according to our experiments reported in Sec. 7, retraction starting from the base complex commonly ends up in worse local minima, i.e., complexes of larger size.

Remark (Sparse Serial Construction) In the 2D case, a not only regular-irreducible but fully irreducible motorcycle graph can be obtained right away by not tracing motorcycles simultaneously but serially, and omitting motorcycles whose neighbors around a singularity have already been traced [EGKT08]§7. This strategy can be applied in the 3D setting as well, as we detail in the supplementary material (part A). However, the reported experiments show that this serial strategy commonly leads to more complex results than wall-retraction applied to the standard simultaneous strategy; we therefore focus on the latter in the following.

5. Implementation

We describe two implementations, one to compute a motorcycle complex of a hexahedral mesh (primarily as an intuitive entry) and one to compute a motorcycle complex of a seamless parametrization on a tetrahedral mesh. In the former case we can exploit that all relevant isosurfaces are available explicitly as facets of hexahedral elements, resulting in a discrete (combinatorial rather than geometric) algorithm; in the latter case isosurfaces arbitrarily cross mesh elements in a continuous manner, requiring additional efforts.

5.1. Mesh-based

In this case the input is a hexahedral mesh, consisting of vertices, edges, facets, and hexes. Implicitly, it has a natural seamless parametrization, mapping hexes to unit cubes. Interior edges are *singular* if their number of incident hexes is different from 4; boundary edges if it is different from 2. For a regular edge e and an incident facet f let $\mathrm{opp}(f,e)$ denote the facet incident to e not incident to a common hex with f; for a boundary edge it may not exist. For an edge e, F_e denotes the set of incident interior facets.

The algorithm makes use of a priority queue Q of (e,f,d) tuples, each with an edge e, a facet f, and a distance $d \in \mathbb{N}$. Queue elements are ordered by d, smallest first.

The condition $\mathrm{alive}(e)$ (line **1**) is true iff at most two facets incident to e are tagged or e is singular. This means that an edge that has already been crossed will not be crossed again (in orthogonal direction). This implies that ties (two fire walls reaching an edge orthogonally with the same distance d) are broken arbitrarily. Note that even if the tie is broken differently on neighboring edges, the resulting partition will be structurally valid (as if both fire fronts had continued), i.e., there is no need for global coordination.

Algorithm 1: Motorcycle Complex of Hexahedral Mesh

```
foreach singular e do Q.push({(e,f,0) | f ∈ F_e}) // ignite
while Q non-empty do
    (e,f,d) ← Q.pop()
1   if alive(e) then   // not crossing burnt terrain
        tag f                    // mark facet as burnt
        foreach regular interior edge e' ≠ e incident to f do
            if opp(e',f) is not tagged then
                Q.push(e', opp(e',f), d+1)        // spread
foreach boundary facet f do tag f
```

Once the algorithm terminates, the union of all tagged facets form the walls of the raw motorcycle complex, partitioning the hexahedral mesh into blocks B_i, each consisting of $m_i \times n_i \times o_i$ hexes

for some $m_i, n_i, o_i \in \mathbb{N}$. The explicit structure and connectivity of the motorcycle complex is then easily discovered by exploiting the connectivity of the underlying hexahedral mesh.

5.2. Parametrization-based

Here the input is a tetrahedral mesh, consisting of vertices, edges, facets, and tets, equipped with a seamless parametrization. In contrast to the algorithm in Sec. 5.1 here we cannot simply walk along the faces of the mesh: the isosurfaces relevant for the motorcycle complex do not coincide with the tetrahedral mesh's facets, but cross its tets arbitrarily. We thus need to perform the brush fire expansion through the interior of tets. Inside each tet the situation can furthermore be highly complex, with multiple fire walls meeting in arbitrary configurations; essentially, within each tet a separate Euclidean 3D motorcycle complex problem is to be dealt with.

We can simplify implementation significantly by refining the mesh on the fly while spreading the fire, so as to have it coincide with facets of the mesh. This simplifies not only the propagation process, but also the representation of the motorcycle complex and the final discovery of its structure and connectivity. The following algorithm spells out this process. Notice the close analogy to Alg. 1, extended to perform and deal with the refinement of the mesh. The choice of the vector n in line **1** is explained in Sec. 5.2.2.

Algorithm 2: Motorcycle Complex of Seamless Parametrization

```
   foreach singular e do                                   // ignite
   |  foreach tet t incident on e do
1  |  |  if f ← iso_facet(e,t) then Q.push(e, f, 0, n)
   while Q non-empty do
   |  (e, f, d, n) ← Q.pop()
   |  if alive(e) then   // not crossing burnt terrain
   |  |  tag f                          // mark facet as burnt
   |  |  foreach regular interior edge e' ≠ e incident to f do
   |  |  |  foreach tet t incident on e' do
2  |  |  |  |  if f' ← iso_facet(e', t, f) ∧ f' not tagged then
3  |  |  |  |  |  Q.push(e', f', d + extent(e, e', n), τn)
   foreach boundary facet f do tag f
```

5.2.1. Mesh Refinement

The method iso_facet(e,t) (line **1**) performs the following: if there is a parametric iso-plane that contains e and intersects the opposite edge e' of t at a point p, the edge e' is split at p, introducing a new vertex v and splitting all incident tets, and the new iso-facet formed by e and v is returned. Otherwise, if any of the two facets of t incident on e is an iso-facet, it is returned. An iso-facet is a facet constant in one of the ϕ-parameter values (u, v, or w). These cases are illustrated in Fig. 10a.

The method iso_facet(e,t,f) (line **2**) behaves as iso_facet(e,t), but considers only iso-facets aligned with f (same constant parameter, taking transitions into account) except f itself.

Additionally, whenever such a split is performed, affected edges and facets in the queue need to be updated. When an edge e is split, each queue entry (e,f) needs to be replaced by two entries (e_0,f_0) and (e_1,f_0), with sub-edges e_0, e_1 and sub-facets f_0, f_1 (see Fig. 10b top). When a facet f is split, but not the edge e of an entry (e,f), it is replaced by (e,f_0), where f_0 is the sub-facet incident on e (see Fig. 10b bottom). A more efficient (slightly more involved) implementation alternative is to postpone these queue updates: We keep a binary forest that records the facet split hierarchy: for each facet that gets split, a record of the two resulting sub-facets is kept. When an entry with facet f and edge e is popped from the queue but f does not exist in the mesh anymore (because it was split), we look up its two children in the hierarchy. Either one or two of these has an edge that is a sub-edge of e (the two cases in Fig. 10b). We push the children *with* an e-sub-edge into the queue and continue. This may proceed recursively, until the sub-elements currently present in the mesh are reached.

A further modification over Alg. 1 is necessary for Alg. 2: The queue is ordered by d only secondarily; primarily, queue entries with e not lying in an original mesh facet are given priority. This ensures that once the brush fire front has entered the space of an original tet, it (atomically) proceeds through this space entirely (i.e. through all refinement-induced sub-tets). This prevents potentially infinite alternating split sequences that could occur when multiple fire walls were spreading inside the same original tet.

In our publicly available implementation, numerical robustness is ensured by representing split vertex coordinates exactly as rational numbers using the GMP library.

5.2.2. Distance Tracking

Compared to the algorithm in Sec. 5.1, in which propagation distances d can quite reasonably be increased in unit steps per hex, here we proceed differently to reduce mesh dependency. The function extent(e,e',n) (line **3**) is defined as follows: Let p_e be the end point of e for which $n^T\phi(p_e)$ is minimal; then we define extent$(e,e',n) = n^T(\phi(p_{e'}) - \phi(p_e))$ (w.r.t. the coordinate chart of facet f). Here n is a unit axis-aligned vector; in the initialization (line **1**) it is orthogonal to the singular edge e and contained in f. During propagation it is transformed using τ (line **3**), the chart transition between f and f' around e'; in this we assume each facet is arbitrarily associated with one of its two adjacent tets, adopting its chart coordinate system.

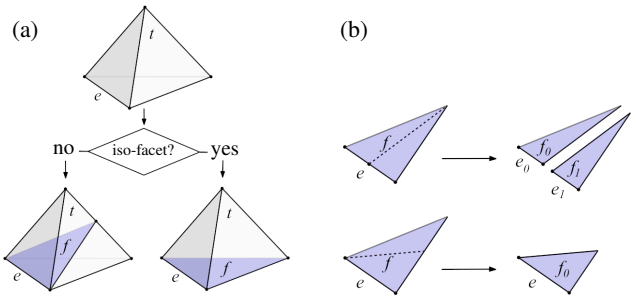


Figure 10: *a) Splitting a tetrahedron along an isoplane using* iso_facet(e,t). *The returned new facet f is marked blue; the special case of an existing iso-facet being returned is also shown. b) Updating a queue entry $(e,f,\cdot,\cdot)$ when e gets split (top), and when f but not e gets split (bottom).*

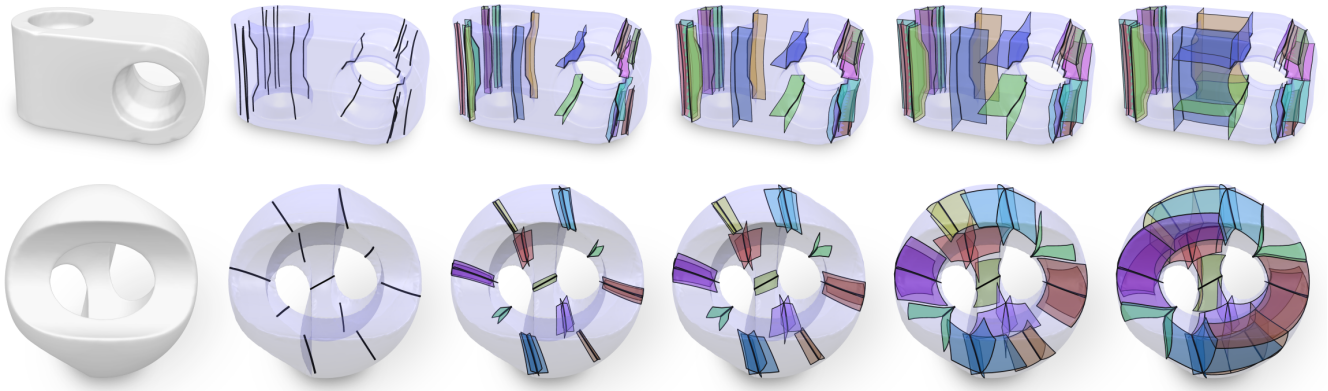


Figure 11: *Snapshots of the motorcycle complex construction algorithm (left to right) in seamless parametrizations of two example objects. The black curves in the second column are the parametrization's singularities, which spawn the fire walls that partition the object's interior. For visual clarity, fire fronts are shown as smooth curves here; the supplementary video shows the raw propagation process in more detail.*

In this way the algorithm measures the parametric travel distance along the conceptual direction of propagation. Note that with this notion of distance, lateral propagation (orthogonal to n) is associated with no increase in distance; this means a fire front that is partially blocked (as in Fig. 5 center) laterally flows around the obstructing wall in a virtually instantaneous manner, rather than forming the circular front conceptually depicted in Fig. 5 center right.

Also note that this implementation performs propagation in a facet-by-facet manner, i.e., fire front collisions are not handled in a continuous manner. In particular, this can lead to non-minimal results. But as non-minimality is an inherent property either way (Sec. 4.2), and as we are therefore going to reduce the resulting complex anyway, the significant added complexity of a continuous collision resolution would be unlikely to be justified in practice.

Fig. 11 illustrates the algorithm on two example models.

5.3. Torus Splitting

In order to turn occasional genus 1 blocks (cf. Fig. 8) into cuboid blocks, one simply detects these (by counting corners) and starts a new brush fire inside the block from an arbitrary point on one of its arcs, confined to the iso-plane that is orthogonal to the two walls incident at that point. This yields an additional wall, cutting the toroidal block to a (self-adjacent) cuboid block.

5.4. Reduction

Following Def. 1, a wall can be removed from the cell complex if the union of its two adjacent blocks is again cuboid and, optionally, it is not incident to a singular arc. This is easily determined: for each of the four arcs surrounding a wall, check that

- the two wall-adjacent blocks form a 90°edge (rather than a 180°edge) each at that arc,
- the two wall-adjacent blocks are actually distinct,
- and optionally: the arc is regular.

Removable walls are queued up, sorted by parametric distance to their origin. This distance is available as value d per facet f during Algorithm 2 and stored accordingly. A wall's distance is defined as the minimum over its facets. We then greedily remove removable walls, starting with the farthest. Whenever a wall is removed, (some of) its arcs may become trivial in the sense that only two incident walls are left; these arcs vanish and the two incident walls are merged. The removability status of all adjacent walls is then retested and the queue updated accordingly.

6. Parametrization Sanitization

Seamless parametrizations are commonly obtained through numerical optimization routines [NRP11]. This involves inaccuracies due to limited precision. The resulting parametrizations therefore commonly are not exactly seamless on a numerical level. This bears some potential of leading to inconsistencies in the construction of the motorcycle complex. For the 2D case, this issue was discussed in detail in previous work [EBCK13, MC19]. The latter article proposes a method that transforms a nearly seamless parametrization of a triangle mesh into one that is truly seamless—while preserving its singularities and boundary alignment. In this section we describe a generalization to the volumetric case on tetrahedral meshes. This enables the safe application of the motorcycle complex algorithm on the resulting truly seamless volumetric parametrization.

6.1. Background: 2D Case

We briefly recapitulate the 2D case, focusing on the differences and referring to the original paper for a complete overview.

The overall constraint system for seamless parametrization consists of chart transition constraints across all non-boundary edges, alignment constraints along the boundary and feature edges, and possibly further constraints like cycle or connection constraints.

Exact Constraint Satisfaction. To obtain an exact solution to the constraint system, close to the given almost-seamless parametrization, [MC19] propose to first separate the variables into two sets—*implied* and *free*—by converting it into integer reduced row echelon form. This can be done without numerical error, confined to the integer domain. This turns the system into upper triangular form, such

that all the implied variables are expressed as linear combinations of free variables with solely rational coefficients. The free variables can then be chosen in such a way that the implied variables can be computed and represented without error using standard floating point arithmetic. To this end, the free variables are initialized according to the values in the given almost-seamless parametrization, but then slightly altered and quantized such that they are divisible by everything they will be divided by in the linear combinations. In this way, the method ensures that ultimately all variables are standard floating point numbers while exactly satisfying all constraints.

This approach is generic and in principle applicable to any homogeneous constraint system (including 3D seamlessness constraints). However, the above constraints form a large system with a size of the same order as the mesh; variables are the (u,v)-parameters of the mesh's vertices. In such cases the approach is impractical. [MC19] showed how a simplified core system (over only certain *sector* variables of particular *node* vertices) can instead be considered, drastically reducing the effective system size. Via generalization, we follow an analogous path for the 3D case.

6.2. 3D Case

The seamless parametrization ϕ consists of linear maps $\phi^t : t \to \mathbb{R}^3$ per tetrahedron t, related across the tetrahedras' facets via transition functions (cf. Sec. 3.1). The transition function across a facet in one direction is the inverse of that across it in the opposite direction.

For our purpose, we are interested in constraints for seamless transitions and boundary alignment being satisfied exactly.

Transition Constraints. Seamlessness can be imposed by requiring for each edge ab of a facet (with *intended* transition function π_{st}) between two tetrahedra s and t:

$$\phi^t(b) - \phi^t(a) = \pi_{st}(\phi^s(b) - \phi^s(a)). \tag{1}$$

Note that only the rotational (not the translational) part of the rigid transformation π_{st} matters in this formulation, as it is applied to vectors rather than points.

Alignment Constraints. For boundary alignment of a facet f of a tetrahedron t, one requires that one particular of the parametrization's three components (u,v,w) is constant along each edge ab of f:

$$\phi^t(b)|_k = \phi^t(a)|_k, \tag{2}$$

where k is 0, 1, or 2, depending on the respective component. To ensure boundary alignment, such a constraint is in effect for all boundary edges.

6.2.1. Terminology

A facet for which the intended transition function is not identity we call a *cut facet*. The union of all cut facets forms the *cut set*. We call an edge a *cut edge* if one of the following holds:

- it is a singularity edge,
- it is incident to one or to more than two cut facets,
- it is a boundary edge and incident to at least one cut facet, or, its incident boundary facets have different alignment.

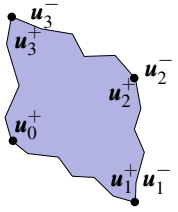
For the matter of this section (for consistency with previous work) we will use the term *node* with a different meaning than in the context of the motorcycle complex in Sec. 4.1. As this section deals with an orthogonal matter and is not concerned with the motorcycle complex, no ambiguities are caused.

We refer to a vertex as *node* if (i) it is incident to one or to more than two cut edges, or (ii) it is a boundary vertex incident to a non-boundary cut edge. A connected set of cut edges bounded by *nodes* form a *branch*. All these branches together partition the cut set and the mesh boundary into pieces we call *sheets*. Each sheet is an orientable 2-manifold surface (with one or multiple boundary loops) and is either a *cut* sheet or an *align* (i.e. boundary) sheet. Note that within each cut sheet, the transition function is constant, and within each boundary sheet, the aligned component (k in Eq. (2)) is constant. Around each vertex, the cut facets partition the tetrahedral mesh into *sectors*, such that all incident tetrahedra within a sector share parametrization values at the vertex.

6.2.2. Simplified Constraint System

The overall constraint system to be satisfied consists of transition constraints across all mesh facets and alignment constraints over boundary facets. In the supplementary material (part B) we show that the sub-system concerned with the non-node vertices can be converted into triangular form, similar to the 2D case [MC19]. It is therefore sufficient to deal with a small core system involving only the variables associated with nodes—a number proportional to the complexity of the singularity structure of ϕ (assuming a sensible cut choice), rather than to the size of the mesh. The proper parametrization values for non-node vertices can easily be computed from the result, in a back-substitution-like manner, afterwards.

More precisely, for this simplified system, we need to consider one $\boldsymbol{u}$-variable per node sector. Note that $\boldsymbol{u} = (u,v,w)$ has three components. For each sheet, we mark one of its nodes as *base* node. In rather rare cases there may be branches which are circular; on these we consider two arbitrary vertices as additional nodes. And there may be loop branches, starting and ending at the same node; on these we consider one arbitrary vertex as additional node. In this way each sheet has at least two nodes on it.



For each sheet (with its marked base node) every other node on it will contribute one equation—either transition or alignment—depending on the type of the sheet. More specifically, given a cut sheet with transition function π and having n nodes with sector variables $\boldsymbol{u}_0^\pm, \boldsymbol{u}_1^\pm, \boldsymbol{u}_2^\pm, \ldots, \boldsymbol{u}_{n-1}^\pm$ ($\pm$ denoting sector variables on the two sides, front and back, of the sheet; see inset figure), the equation corresponding to the i-th node ($0 < i < n$) will be: $\boldsymbol{u}_i^+ - \boldsymbol{u}_0^+ = \pi(\boldsymbol{u}_i^- - \boldsymbol{u}_0^-)$. Similarly, for an align sheet we set up an equation per node: $\boldsymbol{u}_i|_k = \boldsymbol{u}_0|_k$ (no $\pm$-distinction on the boundary).

This simple system is then solved as in [MC19]§5.3.1 ensuring that all the constraints are satisfied while conveniently remaining in the floating point domain. From the result, the precise transition function (including its translational component) of each cut sheet, and the precise constant parameter value of each align sheet is determined (i.e., can be read from the new values at nodes). The

Model	BC	BC^-	raw	MC^+	$\frac{MC^+}{BC}$	T	MC	$\frac{MC}{BC}$
EXAMPLE 3	406136	67828	9087	5780	1.4%	42%	2877	0.7%
EXAMPLE 1	74331	11385	3137	2248	3.0%	15%	1123	1.5%
EXAMPLE 2	3253	678	233	195	6.0%	14%	87	2.7%
DRAGON-HEX	12488	2959	979	724	5.8%	36%	357	2.9%
GARGOYLE	7563	1967	720	546	7.2%	38%	257	3.4%
ANC101 A1	12336	3118	1359	846	6.9%	45%	460	3.7%
FERTILITY-HEX	2002	548	221	189	9.4%	27%	76	3.8%
PEGASUS-HEX	9745	2415	1035	729	7.5%	36%	374	3.8%
KISS HEX	5019	1194	543	385	7.7%	39%	200	4.0%
ANC101	5009	1283	609	347	6.9%	39%	207	4.1%
IMPELLER STRESSTEST	878	176	184	124	14.1%	24%	37	4.2%
ARMADILLO HEX-A	5960	1491	680	516	8.7%	34%	266	4.5%
ARMADILLO HEX-B	3265	820	396	296	9.1%	31%	147	4.5%
			⋮					
EXAMPLE 5	1	1	1	1	100%	0%	1	100%

Table 1: *Statistics on a dataset of hexahedral meshes (full table in supplementary material). Reported are numbers of blocks in the base complex (BC), reduced base complex (BC^-), raw motorcycle complex (raw), reduced motorcycle complex with preserved singularity-adjacent walls (MC^+), and fully reduced motorcycle complex (MC) – ordered by complexity of MC relative to BC. It can be observed that the raw MC is typically larger than the MC^+ (or MC) by a factor of around 1.4 (or 2.9) only, i.e., construction overhead over a hypothetical direct construction of the final MC is benign. Furthermore, notice that the fully reduced BC^- is generally significantly larger than the fully reduced MC (see the remark in Sec. 4.2). On average one third of the MC's arcs are T-arcs (T).*

Model	tets	BC	MC	$\frac{MC}{BC}$	facets	trace	build	reduce
ROCKERARM	97 K	2446	78	3%	154 K	27.9 s	11.1 s	1.9 s
ARMADILLO	163 K	3110	132	4%	260 K	46.4 s	16.8 s	1.6 s
JOINT	42 K	205	15	7%	56 K	9.5 s	3.6 s	0.6 s
KITTEN	30 K	208	19	9%	49 K	8.3 s	3.4 s	0.7 s
BROKEN BULLET	20 K	44	5	11%	13 K	2.1 s	0.9 s	0.1 s
SCULPTURE	20 K	108	13	12%	17 K	2.8 s	1.2 s	0.1 s
FANDISK	46 K	128	19	15%	38 K	6.6 s	2.5 s	0.3 s
BONE	54 K	87	15	17%	45 K	7.4 s	3.0 s	0.8 s
CAMILLE HAND	103 K	142	26	18%	74 K	12.5 s	5.6 s	1.2 s
CYLINDER	12 K	26	5	19%	14 K	2.4 s	0.9 s	0.1 s
SPHERE	19 K	7	2	29%	6 K	0.9 s	0.4 s	0.1 s
CUBE SPHERE	11 K	10	4	40%	4 K	0.6 s	0.3 s	0.0 s
TETRAHEDRON	14 K	4	2	50%	2 K	0.3 s	0.2 s	0.0 s
FANPART	5 K	5	3	60%	2 K	0.3 s	0.1 s	0.0 s
PRISMA	21 K	3	2	67%	3 K	0.5 s	0.4 s	0.0 s

Table 2: *Statistics on a dataset of seamless parametrizations of tetrahedral meshes. Columns show the number of tetrahedra in the input meshes (*tets*), the number of triangular mesh facets tagged by Alg. 2 (*facets*), and the time spent in the three algorithmic steps (tracing the fire walls, building a graph representation of the complex (including torus splitting), and wall retraction for reduction). Notice that, similar to Table 1, the BC again has up to* 30× *as many blocks as the MC of the same model.*

parametrization values of each non-node sheet vertex can then easily be updated to match this: fix the $\boldsymbol{u}$-value in one sector (after appropriate precision truncation, and possibly taking the determined alignment value into account), and propagate it to the other sectors of this vertex using the determined transition functions. At non-node vertices on *singular* branches a little extra care is required: the composed transitions around the branch imply fixpoint parameters [EBCK13]§3.2; these need to be chosen, so as to keep the propagation consistent across all sectors.

Afterwards, the parametrization is exactly seamless and exactly boundary-aligned, such that the motorcycle complex construction algorithm from Sec. 5.2 can be applied safely.

7. Results

In the following we evaluate the characteristics of the motorcycle complex, in particular in comparison to the base complex, as well as the proposed algorithms (mesh-based and parametrization-based) for its construction. The supplementary video provides several animated impressions of the process. Hexahedral mesh visualizations are rendered using [BTP*19].

Mesh-based Algorithm

We take a dataset of 261 all-hex meshes, collected by [BTP*19], generated by a variety of hexahedral meshing approaches, e.g. [FXBH16, LSVT15, LLX*12, GSZ11], and apply our mesh-based algorithm (Sec. 5.1). Table 1 provides statistics on the results, including the motorcycle partitions' size before and after reduction (Sec. 4.2). The full table is available in the supplementary material.

An interesting comparison is with respect to the standard base complex. We include the corresponding statistics in Table 1. As can be observed, the motorcycle complex is often simpler by a large factor (here up to 140×). Besides having an obvious positive effect on construction cost, the motorcycle complex offers benefits on the application side, as demonstrated in Secs. 7.1 and 7.2. Splitting of toroidal blocks (Sec. 5.3) occurred in 13 of the models, a total of 48 times.

Parametrization-based Algorithm

We apply the parametrization-based algorithm (Sec. 5.2) to seamless parametrizations on tetrahedral meshes, generated using frame field guided parametrization, i.e., by solving Eq. (10) from [NRP11] (without integer constraints, without rounding). In this we use frame fields provided by the authors of [LZC*18], corresponding to the results shown in that article. Numerical sanitization of these parametrizations (Sec. 6) took less than a second for most cases, 7s for the most complex case (with 163K tets). Table 2 shows details about these runs, including the number of facets traversed and forming the motorcycle complex walls. Notice that again the base complex is significantly more complex; the number of blocks is up to 30× higher, construction time up to 13×, memory consumption up to 9×.

Remark: The motorcycle complex is well-defined only for *valid* seamless parametrizations in general. Their fully robust generation in 3D is a problem under broad investigation, following recent advances regarding the analogous 2D problem [ZTZC20, CSZZ19, CCS*21]. In particular because our algorithm operates on generic *continuous* rather than special *quantized* parametrizations, it was easy, though, to yield valid input parametrizations for 15 out of 19 models from [LZC*18] already with the above simple best-effort approach following [NRP11].

7.1. Example Use Case: Quantization for Hex Meshing

A *continuous* seamless parametrization, as discussed in Sec. 3.1, can be viewed as defining an infinitely fine hexahedral mesh, whereas the special case of a *quantized* seamless parametrization (also called integer-grid map) implies a finite hexahedral mesh. In the 2D case, this relation is exploited in state-of-the-art quadrilateral mesh generation methods. [KNP07] pioneered the idea of first generating a continuous seamless parametrization, and then *rounding* it (at once or iteratively [BZK09]) to a quantized seamless parametrization. This rounding is a notoriously fragile process, though: With increasing target quad size, the risk of yielding an invalid parametrization (with degenerate or flipped parts) increases, as pointed out and demonstrated in Fig. 1 of [BCE*13] and Fig. 3 of [CBK15]. An analogous rounding procedure has been described for the 3D case [NRP11]; it is the state-of-the-art approach to yield quantized volumetric seamless parametrizations, as evidenced by its sustained use in recent works [FXBH16, SVB17, LZC*18, CC19, PBS20]. Not surprisingly it comes with the same limitations as its 2D counterpart, as also evidenced in Table 3.

In the 2D case, subsequent work has provided a remedy, taking a different path, reliable and efficient, from continuous to quantized seamless parametrizations: via the motorcycle graph [CBK15, LCBK19, LCK21a]; for the 3D case, this path has not been paved yet. Our motorcycle complex is the key to extending this state-of-the-art approach to the 3D case, generating hexahedral meshes (that are preferred over tetrahedral meshes for certain use cases [Bla01, SRRGRN14]) via volumetric seamless parametrizations.

As a proof of concept, to illustrate the potential, we translate a simple version of this approach to the 3D setting: we compute the motorcycle complex of a continuous seamless parametrization, and then scale the parametrization within each block such that it adopts integer dimensions, trivially implying some $l \times m \times n$-grid of unit hexahedra per block. The dimensions and the scaling need to be chosen such that these grids conform across block boundaries. This is achieved by expressing the quantization (the integer dimensions choice) by assigning integer lengths to arcs—shared between walls and blocks, inherently ensuring compatibility.

What we need to require for this assignment is that walls remain rectangles (thus blocks remain rectangular cuboids) parametrically. Let A_i, $i \in \{0,1,2,3\}$, denote the set of arcs forming the four sides of a wall, and $\ell_a \in \mathbb{Z}^{>0}$ the length assignment of arc a. Then this requirement can be expressed using two linear constraints per wall:

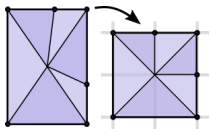
$$\sum_{a \in A_i} \ell_a = \sum_{a \in A_{i+2}} \ell_a, \quad i = 0,1 \tag{3}$$

One aims to reproduce the sizing of the given parametrization using

$$\sum_a (\ell_a - s\|a\|)^2 \to \min, \tag{4}$$

where $\|a\|$ denotes the original parametric length, and s is a scaling factor that allows choosing the resulting mesh's resolution.

Note that the parametrization per block cannot be scaled by a simple affine map as each of the block's six facets consists of possibly multiple walls (due to T-joints from outside the block), and each wall needs to be scaled according to its arcs' values ℓ_a. It can be achieved via a *piecewise*-affine map σ, though: affine per tetrahedron spanned by the block's center point with a triangle in a conforming triangulation of the (rectangular) walls on its surface. The inset illustrates a 2D version. If one splits the underlying tetrahedral mesh by these meta-tetrahedras' faces, σ is affine per element and no inversions occur under $\sigma \circ \phi$ (most of this refinement is superfluous and can be omitted). A smoothing of either the resulting parametrization [RPPSH17] or the implied hex mesh [LSVT15] can be applied subsequently to distribute distortion evenly.



Model	MC	$\frac{\text{MC}}{\text{Round}}$	Round	inverted	HexEx
KITTEN	176	5.7%	3112	9.1%	broken
ARMADILLO	1884	6.2%	30456	7.3%	broken
CAMILLE HAND	122	8.5%	1438	5.6%	broken
BONE	87	13.9%	628	4.8%	broken
SCULPTURE	108	16.8%	642	1%	valid
SPHERE	7	21.9%	32	5.3%	valid
FANDISK	110	21.9%	502	1.3%	valid
JOINT	205	23.1%	888	1.6%	broken
ROCKERARM	1391	29.1%	4784	2.7%	broken
PRISMA	3	33.3%	9	2.3%	valid
FANPART	5	38.5%	13	0.6%	valid
CYLINDER	26	43.3%	60	3.6%	valid
CUBE SPHERE	10	100.0%	10	0%	valid
TETRAHEDRON	4	100.0%	4	0%	valid
BROKEN BULLET	44	122.2%	36	0%	valid
STAR BOLT	-	-	-	0.1%	valid
KNOT	-	-	-	0.9%	broken
BUNNY	-	-	-	0.1%	broken
ELEPHANT	-	-	-	0.2%	broken

Table 3: *Statistics on the maximum coarseness (number of hexes) of hexahedral meshes generated from seamless parameterizations using our approach employing the motorcycle complex (MC), and via the classical method of iterative rounding (Round). We also report the percentage of parametrically inverted (or degenerate) tetrahedra when trying to achieve the coarseness of MC (or a very fine mesh in the four bottom rows) with the rounding-based approach. As can be seen in the last column, the HexEx approach from [LBK16] is able to recover a valid hex mesh from these invalidly rounded parametrizations only in mild cases. Also see Fig. 16.*

Comparison to Rounding To give an idea of the benefit, in Table 3 we compare this motorcycle complex based quantization strategy with the classical rounding strategy on a dataset of 19 tetrahedral meshes with frame fields, namely those shown in [LZC*18]. It can be seen that for 15 of these a valid continuous seamless parametrization can be obtained by solving Eq. (10) from [NRP11] to begin with—namely those used for the above experiments in Table 2. To these 15 continuous parametrizations we applied the rounding strategy, increasing the target edge length (via sizing s) until failure, and state the number of hexahedra implied by the coarsest rounded seamless parametrization that could validly be obtained. We also applied the motorcycle complex based quantization strategy, and state the number of hexahedra obtained when setting $s = 0$ (aiming for maximal coarseness) in Eq. (4). It can be

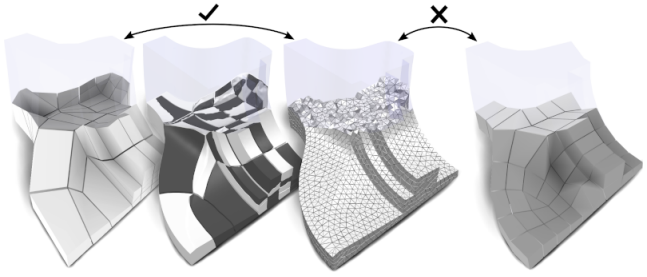


Figure 12: *Left: The quantization approach yields a hex mesh (left) with volumetric correspondence between each hex and a region (checkerboard piece) of the input object. Right: A hex mesh recovered by HexEx does not come with such a volumetric map.*

observed that, except for the simplest models, typically a significantly coarser quantization, thus coarser hex mesh can be obtained.

We remark that HexEx [LBK16] can sometimes extract valid hex meshes even from invalid rounded parametrizations. We applied it to parametrizations rounded to the same level of coarseness as can be achieved with our approach. In 7 of the 16 cases where the rounded parametrization is invalid, it was able to output a valid hex mesh; in 9 cases the hex mesh has defects, some examples of which are shown in Fig. 16. Note that when HexEx succeeds in ignoring the parametrization's defects, it does output a mesh but, in contrast to our approach, no valid parametrization, in particular no bijection between the input and the output mesh (Fig. 12).

Alternative: Coarsening. One may consider the alternative of only applying mild (therefore more robust) rounding, yielding an overly fine initial hex mesh, followed by coarsening, e.g., using [GPW*17]. Obvious downsides are the lack of a priori knowledge of a successful target edge length setting (potentially requiring trial-and-error) as well as the higher time and memory cost of this approach, operating fine-to-coarse rather than coarse-to-fine. Furthermore, the conforming sheet operators employed for structure and singularity preserving mesh coarsening are more restricted than a motorcycle complex based quantization procedure, cf. Fig. 13.

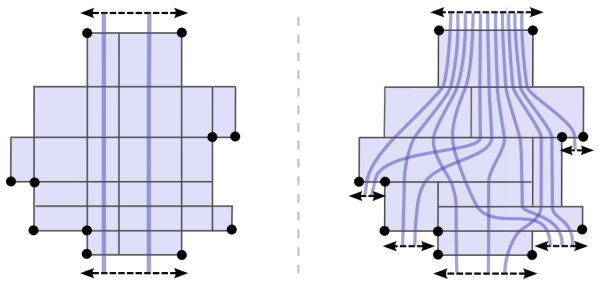


Figure 13: *Illustration of a part of a 2D slice through a BC (left) or MC (right); black dots are singularities. Assume the distance (parametric or number of hexes) between the upper two singularities shall be increased or decreased (in a quantization or a hex mesh). In the BC, this can be achieved using one of two* sheet operators *(purple paths), here with essentially identical effect. In the MC, there are 11 different paths to adjust the quantization, with a choice of different effects on other singular, boundary, or feature points.*

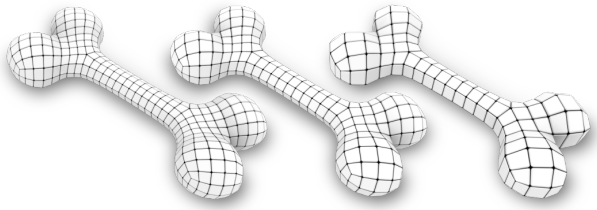


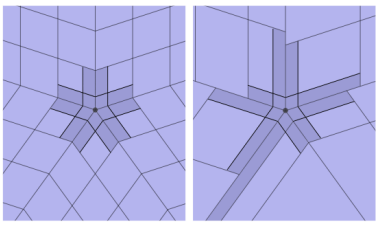
Figure 14: *Hexahedral meshes of varying density obtained by quantizing a volumetric seamless parametrization using the proposed motorcycle complex.*

Alternative: Base Complex. For the same reason (in addition to complexity-related reasons), it is better to formulate the quantization problem (3)+(4) based on the coarser and non-conforming motorcycle complex than on the base complex: there are more degrees of freedom, more ways for the solver to adjust the quantization length assignment ℓ (while respecting (3)), as illustrated in Fig. 13. This in particular enables fine-grained control over the resulting mesh sizing (Fig. 14; also see supplemental material part D).

We point out that, while these experiments already demonstrate various benefits, further improvements are possible and shall be explored in future work. For instance, we solve the integer program (3)+(4) using a general purpose solver (Gurobi); a tailored strategy, along the lines of [CBK15], could be more efficient. Block reparametrization (to match the quantization) could be performed using efficient combinations of fast (e.g. discrete harmonic mapping) and reliable fallback (e.g. the piecewise-affine σ) solutions. For simplicity, we required $\ell_a > 0$; supporting zero-arcs requires additional efforts [LCBK19] but will enable higher quality.

7.2. Example Use Case: Solid T-Splines

T-splines are a flexible tool in the context of smooth function representation, for geometric modelling as well as for isogeometric



Model	+BC	+MC	factor
1	12743	4190	3.0×
2	28352	14656	1.9×
3	14520	10736	1.4×
4	31217	15407	2.0×
5	17866	10608	1.7×
6	26785	14167	1.9×
7	14520	7348	2.0×
8	11165	6437	1.7×
9	24685	12720	1.9×
10	19030	11303	1.7×

Figure 15: *T-spline-required refinement of blocks around a singular arc (cross section view) in a conforming (left) versus a non-conforming (e.g. MC) complex (center). Right: The T-mesh derived in this way from the MC is typically much coarser compared to the BC; on 10 example models from Table 1, the number of additional refinement-induced cells (+BC and +MC, respectively) is 1.4-3.0 times larger for the (already initially larger) BC.*

Figure 16: *Hexahedral meshes of increasing coarseness obtained from seamlessly parametrized tetrahedral meshes by means of iterative rounding to an integer-grid map as described by [NRP11] (dark gray) and our method using the motorcycle complex (white). When increasing target coarseness, hexahedral meshes obtained by rounding become increasingly defective due to parametric degenerations and inversions. Even fault-tolerant mesh extraction [LBK16], as employed here, cannot recover from this, leaving gaps that cannot easily be patched.*

analysis. For the volumetric case, constructions of T-spline spaces starting from hexahedral meshes have been described [WZXH12]. As solid T-splines are defined over cuboid complexes which are not necessarily conforming (hence the 'T'), it is actually unnecessarily restrictive to start from a conforming one, i.e., a hexahedral mesh. We can essentially apply the necessary structural refinement around singularities that the above paper describes directly on the non-conforming motorcycle complex. This circumvents the need for quantization (to obtain a hex mesh), and effectively provides a coarser starting configuration—which could then be adaptively refined where necessary for a particular application, as opposed to starting with a rather dense hexahedral mesh as domain structure and later coarsening it where possible.

In Fig. 15 we illustrate the refinement around a singularity (inserting additional walls) by the rules of [ZWH12], so as to yield a T-mesh suitable as control mesh for a solid T-spline. We note that some additional modifications to the control mesh structure can be necessary to ensure continuity due to non-local overlaps of basis function supports with singularities, as discussed in [CZ17]§8.2, or to yield specific classes of splines, such as analysis-suitable splines, as discussed in [SLSH12]. In any case, the motorcycle complex provides a significantly simpler starting point than the base complex (or a hexahedral mesh derived from it). We contrast the numbers of additional T-mesh blocks due to refinement-at-singularities applied to the BC and the MC in Fig 15.

8. Conclusion

We have introduced a generalization of the motorcycle graph to the volumetric setting, providing a structure and algorithm that enable the compact block decomposition of solids. Hexahedral meshes or seamless volume parametrizations can serve as underlying basis. We expect the 3D motorcycle complex will enable progress in various ways, just like the 2D original has found effective use in the context of parametrization, mesh generation, and mesh processing, in particular when it comes to providing robustness guarantees. We have demonstrated that our generalization has the potential to form the basis for extensions of such techniques to the (even more relevant) 3D cases where such guarantees are still lacking.

Limitations & Future Work

While we have demonstrated that the motorcycle complex typically is very small, it is not necessarily the smallest non-conforming cuboid partition. Finding the actual minimum partition is very hard already in the 2D case [EGKT08]. It would be interesting (even if not necessarily of high practical relevance) to investigate whether the size of the motorcycle complex relative to the minimal size is, as in 2D, bounded in some nice manner. The variation of propagation speed could, as in 2D [GMSO14], enable further size reduction.

In our prototype implementation we employ a very generic polyhedral mesh data structure (OpenVolumeMesh [KBK13]). In this case the computation of the complex is dominated by the tetrahedral splits (around 0.3ms per split on a commodity PC). A tailored lightweight data structure could likely reduce this.

We have demonstrated the use case of quantization (Sec. 7.1), where the motorcycle complex can serve as key ingredient in the process of hexahedral mesh generation. Further developments in this direction, e.g., additionally enabling zero-quantizations for coarse meshes, specializing solvers for efficiency, are of high relevance for ongoing developments in the field of mesh generation.

Some form of generalization to hex-dominant meshes, analogous to the 2D case [SPGT18], could furthermore be of interest. This comes with additional challenges due to the greater structural variability compared to quad-dominant meshes. For the parametrization-based algorithm, an extension to high-order parametrizations [MC20] could be interesting, and the incorporation of some form of tolerance to defects (degeneracies, local inversions) would broaden practical applicability. This latter direction has not even been explored for the 2D case yet, but insights from fault-tolerant mesh extraction [EBCK13, LBK16] may provide inspiration. Another way around such difficulties could be the definition and generation of a motorcycle complex like structure based on frame fields rather than parametrizations, as proved possible and useful in the 2D case [MPZ14], though this comes with major additional challenges in 3D [SOG*21].

Acknowledgments

The authors wish to thank David Bommes and Heng Liu for support with comparisons, Florian Janosch for his preliminary exploration of a discrete implementation that has been helpful for parts of this work. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) - 427469366. Open Access funding enabled and organized by Projekt DEAL.

References

[BCE*13] Bommes D., Campen M., Ebke H.-C., Alliez P., Kobbelt L.: Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4 (2013). 3, 11

[Bla01] Blacker T.: Automated conformal hexahedral meshing constraints, challenges and opportunities. *Engineering with Computers 17*, 3 (2001), 201–210. 11

[BLK11] Bommes D., Lempfer T., Kobbelt L.: Global structure optimization of quadrilateral meshes. *Computer Graphics Forum 30*, 2 (2011), 375–384. 2, 3

[BTP*19] Bracci M., Tarini M., Pietroni N., Livesu M., Cignoni P.: Hexalab.net: An online viewer for hexahedral meshes. *Computer-Aided Design 110* (2019), 24–36. 10

[BZK09] Bommes D., Zimmer H., Kobbelt L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009), 77:1–77:10. 2, 11

[CBK15] Campen M., Bommes D., Kobbelt L.: Quantized global parametrization. *ACM Trans. Graph. 34*, 6 (2015). 2, 3, 11, 12

[CC19] Corman E., Crane K.: Symmetric moving frames. *ACM Trans. Graph. 38*, 4 (2019). 3, 11

[CCS*21] Campen M., Capouellez R., Shen H., Zhu L., Panozzo D., Zorin D.: Efficient and robust discrete conformal equivalence with boundary. *ACM Trans. Graph. 40*, 6 (2021). 10

[CK14] Campen M., Kobbelt L.: Quad layout embedding via aligned parameterization. *Comp. Graph. Forum 33*, 8 (2014), 69–81. 2

[CSZZ19] Campen M., Shen H., Zhou J., Zorin D.: Seamless parametrization with arbitrary cones for arbitrary genus. *ACM Trans. Graph. 39*, 1 (2019). 10

[CZ17] Campen M., Zorin D.: Similarity maps and field-guided T-splines: a perfect couple. *ACM Trans. Graph. 36*, 4 (2017), 1–16. 2, 3, 13

[EBCK13] Ebke H.-C., Bommes D., Campen M., Kobbelt L.: Qex: robust quad mesh extraction. *ACM Trans. Graph. 32*, 6 (2013). 3, 8, 10, 14

[EE99] Eppstein D., Erickson J.: Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete & Comp. Geometry 22*, 4 (1999), 569–592. 1, 2

[EGKT08] Eppstein D., Goodrich M. T., Kim E., Tamstorf R.: Motorcycle graphs: canonical quad mesh partitioning. *Comp. Graph. Forum 27*, 5 (2008). 1, 2, 3, 4, 6, 14

[FXBH16] Fang X., Xu W., Bao H., Huang J.: All-hex meshing using closed-form induced polycube. *ACM Trans. Graph. 35*, 4 (2016). 10, 11

[GDC15] Gao X., Deng Z., Chen G.: Hexahedral mesh re-parameterization from aligned base-complex. *ACM Trans. Graph. 34*, 4 (2015). 3

[GMSO14] Gunpinar E., Moriguchi M., Suzuki H., Ohtake Y.: Motorcycle graph enumeration from quadrilateral meshes for reverse engineering. *Computer-Aided Design 55* (2014), 64–80. 2, 14

[GPW*17] Gao X., Panozzo D., Wang W., Deng Z., Chen G.: Robust structure simplification for hex re-meshing. *ACM Trans. Graph. 36*, 6 (2017). 12

[GSZ11] Gregson J., Sheffer A., Zhang E.: All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum 30*, 5 (2011). 10

[HTWB11] Huang J., Tong Y., Wei H., Bao H.: Boundary aligned smooth 3d cross-frame field. *ACM Trans. Graph. 30*, 6 (2011). 3

[KBK13] KREMER M., BOMMES D., KOBBELT L.: OpenVolumeMesh–a versatile index-based data structure for 3D polytopal complexes. In *Proceedings of the 21st International Meshing Roundtable*. Springer, 2013, pp. 531–548. 14

[KNP07] KÄLBERER F., NIESER M., POLTHIER K.: QuadCover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (2007). 2, 11

[KPP17] KARČIAUSKAS K., PANOZZO D., PETERS J.: T-junctions in spline surfaces. *ACM Trans. Graph. 36*, 5 (2017). 3

[LBK16] LYON M., BOMMES D., KOBBELT L.: HexEx: Robust hexahedral mesh extraction. *ACM Trans. Graph. 35*, 4 (2016). 11, 12, 13, 14

[LCBK19] LYON M., CAMPEN M., BOMMES D., KOBBELT L.: Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Trans. Graph. 38*, 4 (2019). 2, 11, 12

[LCK21a] LYON M., CAMPEN M., KOBBELT L.: Quad layouts via constrained t-mesh quantization. *Computer Graphics Forum 40*, 2 (2021). 2, 11

[LCK21b] LYON M., CAMPEN M., KOBBELT L.: Simpler quad layouts using relaxed singularities. *Comp. Graph. Forum 40*, 5 (2021). 2

[LLX*12] LI Y., LIU Y., XU W., WANG W., GUO B.: All-hex meshing using singularity-restricted field. *ACM Trans. Graph. 31*, 6 (2012). 10

[LLZ*20] LIU H.-Y., LIU Z.-Y., ZHAO Z.-Y., LIU L., FU X.-M.: Practical fabrication of discrete chebyshev nets. *Comp. Graph. Forum 39*, 7 (2020). 2

[LPP*20] LIVESU M., PIETRONI N., PUPPO E., SHEFFER A., CIGNONI P.: Loopycuts: Practical feature-preserving block decomposition for strongly hex-dominant meshing. *ACM Trans. Graph. 39*, 4 (2020). 3

[LSVT15] LIVESU M., SHEFFER A., VINING N., TARINI M.: Practical hex-mesh optimization via edge-cone rectification. *ACM Trans. Graph. 34*, 4 (2015). 10, 11

[LZC*18] LIU H., ZHANG P., CHIEN E., SOLOMON J., BOMMES D.: Singularity-constrained octahedral fields for hexahedral meshing. *ACM Trans. Graph. 37*, 4 (2018). 3, 4, 5, 10, 11

[MC19] MANDAD M., CAMPEN M.: Exact constraint satisfaction for truly seamless parametrization. *Computer Graphics Forum 38*, 2 (2019), 135–145. 8, 9

[MC20] MANDAD M., CAMPEN M.: Efficient piecewise higher-order parametrization of discrete surfaces with local and global injectivity. *Computer-Aided Design 127* (2020), 102862. 14

[MPZ14] MYLES A., PIETRONI N., ZORIN D.: Robust field-aligned global parametrization. *ACM Trans. Graph. 33*, 4 (2014). 2, 14

[MZ12] MYLES A., ZORIN D.: Global parametrization by incremental flattening. *ACM Trans. Graph. 31*, 4 (2012), 109:1–109:11. 2, 3

[NHE*19] NUVOLI S., HERNANDEZ A., ESPERANÇA C., SCATENI R., CIGNONI P., PIETRONI N.: Quadmixer: layout preserving blending of quadrilateral meshes. *ACM Trans. Graph. 38*, 6 (2019), 1–13. 2

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: Cubecover – parameterization of 3d volumes. *Comp. GRaph. Forum 30*, 5 (2011), 1397–1406. 3, 8, 10, 11, 13

[PBS20] PALMER D., BOMMES D., SOLOMON J.: Algebraic representations for volumetric frame fields. *ACM Trans. Graph. 39*, 2 (2020). 11

[PPM*16] PIETRONI N., PUPPO E., MARCIAS G., SCOPIGNO R., CIGNONI P.: Tracing field-coherent quad layouts. *Computer Graphics Forum 35*, 7 (2016). 2

[RP17] RAZAFINDRAZAKA F., POLTHIER K.: Optimal base complexes for quadrilateral meshes. *Computer Aided Geometric Design 52* (2017), 63–74. 2

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 4 (2017). 11

[SERB99] SHEFFER A., ETZION M., RAPPOPORT A., BERCOVIER M.: Hexahedral mesh generation using the embedded voronoi graph. *Eng. with Comput. 15*, 3 (1999). 3

[SKO*10] STATEN M. L., KERR R. A., OWEN S. J., BLACKER T. D., STUPAZZINI M., SHIMADA K.: Unconstrained plastering. *Int. J. Num. Methods Eng. 81*, 2 (2010). 3

[SLSH12] SCOTT M. A., LI X., SEDERBERG T. W., HUGHES T. J.: Local refinement of analysis-suitable t-splines. *Comp. Meth. Appl. Mech. Eng. 213* (2012). 13

[SOG*21] STUTZ F. C., OLSEN T. F., GROEN J. P., AAGE N., SIGMUND O., SOLOMON J., BÆRENTZEN J. A.: Synthesis of frame field-aligned multi-laminar structures. *arXiv preprint arXiv:2104.05550* (2021). 14

[SPGT18] SCHERTLER N., PANOZZO D., GUMHOLD S., TARINI M.: Generalized motorcycle graphs for imperfect quad-dominant meshes. *ACM Trans. Graph. 37*, 4 (2018). 2, 14

[SRRGRN14] SARRATE RAMOS J., RUIZ-GIRONÉS E., ROCA NAVARRO F. J.: Unstructured and semi-structured hexahedral mesh generation methods. *Comp. Tech. Reviews 10* (2014), 35–64. 11

[SVB17] SOLOMON J., VAXMAN A., BOMMES D.: Boundary element octahedral fields in volumes. *ACM Trans. Graph. 36*, 4 (2017). 3, 11

[Tak19] TAKAYAMA K.: Dual sheet meshing: An interactive approach to robust hexahedralization. *Comp. Graph. Forum 38*, 2 (2019), 37–48. 3

[VCD*16] VAXMAN A., CAMPEN M., DIAMANTI O., PANOZZO D., BOMMES D., HILDEBRANDT K., BEN-CHEN M.: Directional field synthesis, design, and processing. *Computer Graphics Forum 35*, 2 (2016). 2

[WZXH12] WANG W., ZHANG Y., XU G., HUGHES T. J. R.: Converting an unstructured quadrilateral/hexahedral mesh to a rational t-spline. *Comput. Mech. 50*, 1 (2012). 13

[ZTZC20] ZHOU J., TU C., ZORIN D., CAMPEN M.: Combinatorial construction of seamless parameter domains. *Computer Graphics Forum 39*, 2 (2020), 179–190. 10

[ZVC*20] ZHANG P., VEKHTER J., CHIEN E., BOMMES D., VOUGA E., SOLOMON J.: Octahedral frames for feature-aligned cross fields. *ACM Trans. Graph. 39*, 3 (2020). 3

[ZWH12] ZHANG Y., WANG W., HUGHES T. J.: Solid t-spline construction from boundary representations for genus-zero geometry. *Comput. Meth. Appl. Mech. Eng. 249-252* (2012). 13