EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Volume Quantization with Flexible Singularities for Hexahedral Meshing

H. Brückler and M. Campen

Paderborn University, Germany

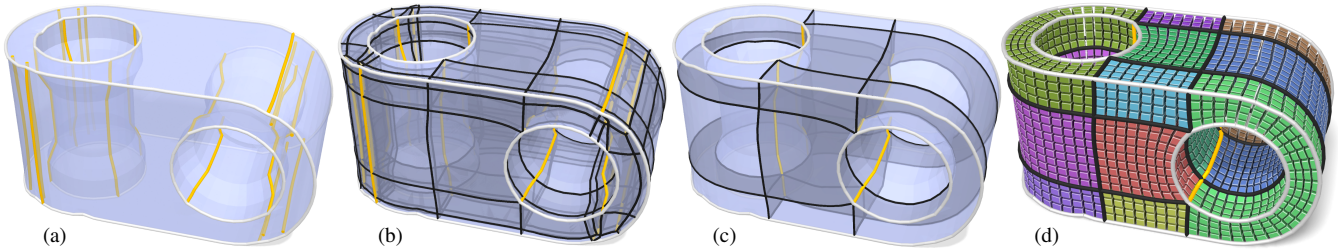


Figure 1: *A state-of-the-art approach to hexahedral mesh generation is via seamless volume parametrization, with predetermined singularities (yellow, here from [CAS*19]) (a). Adhering to these predetermined singularities can result in complex and distorted structures with thin sheets, as demonstrated here with the coarsest conforming base complex of the domain (black) (b). Our novel quantization method regards the given singularities as flexible guidance only and enables implicit simplification wherever beneficial for the targeted mesh resolution (c) – all while guaranteeing that topology and features (white curves) are preserved exactly. This results in more neatly structured hexahedral meshes (d).*

Abstract

We present a novel algorithm for quantization and subsequent hexahedral mesh generation from seamless volumetric maps. Quantization is the process of choosing integers that represent the numbers of hexahedral elements to be placed in each region of the volume, and transforming the seamless map into an integer-grid map matching that choice, inducing a hexahedral mesh. Previous work computes such quantizations under the restriction of a fixed predetermined singularity graph. Our novel approach allows for implicit modification and, in particular, simplification of the map's singularity structure wherever that benefits the chosen objective, such as matching target hexahedron sizes as closely as possible. It comes with two novel ingredients: A feature-focused distortion measure guiding the quantization, and constraints ensuring map injectivity and structure preservation of geometric and topological features, both without relying on a fixed singularity structure. We demonstrate the benefit of the added flexibility offered by this approach: it allows for the generation of hexahedral meshes that more accurately match a desired resolution globally, as well as of meshes exhibiting a simpler block structure.

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

Volumetric meshes, as discretizations of 3D domains, are the very foundation of many numerical methods such as the finite element method (FEM), finite volume method (FVM), discontinuous Galerkin method (DG), or isogeometric analysis (IGA) . The two main choices for the basic elements of these meshes are tetrahedra or hexahedra. Compared to unstructured tetrahedral meshes, the properties of (semi)structured pure hexahedral meshes may yield a number of benefits depending on the the application [SRRGRN14, CK92, BPM*95, RS06, BTPB07, TEC11, Bla01, WCO21, WNR04], but at the same time make them much harder to generate.

To make the hard problem of general and robust hexahedral meshing tractable, approaches tend to adopt the following relaxations to varying extents – sometimes explicitly, sometimes implicitly:

A Widening the result space, e.g., by allowing for some non-hexahedral elements or non-exact feature alignment,
B Shrinking the result space, e.g., by targetting only a subset of possible hexahedral meshes such as polycube meshes,
C Optimizing and fixing degrees of freedom sequentially rather than all at the same time, neglecting their interdependence at the potential cost of result quality.

Our approach follows the frame-field/integer-grid-map pipeline,

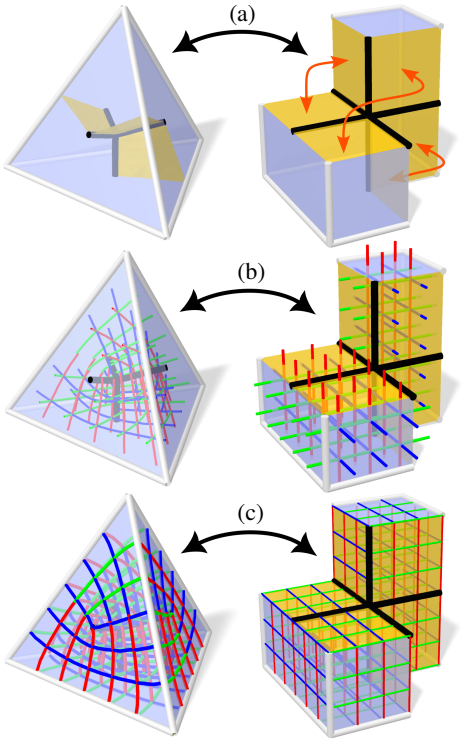


Figure 2: *(a) Object space (left) and parameter space (right) of a general seamless parametrization with prescribed feature curves (white), singularities (black) and corresponding cut surface (yellow); cut-surface patches are linked via rotations from the octahedral group (orange), features and singularities lie along principal parametric axes. (b) The parametric integer grid and its preimage under the general* seamless *map. (c) The parametric integer grid and its preimage under a more strict* integer-grid *map, with integer-alignment of features and singularities.*

which receives increased attention in recent scientific literature [PCS*22, BC23, LB23, BBC24, VCV25, KHB25] as it is able to avoid **A** and **B** completely, but on the flipside relies strongly on **C**. Our main goal is to mitigate the negative impact of the latter, by allowing a crucial degree of freedom, the singularity structure, to change during later stages of the pipeline where previous works have assumed it to be fixed.

While the dynamic evolution of the field makes a canonical formulation of said pipeline difficult, the following core steps can generally be identified:

1) Tetrahedralization of the domain.
2) Computation of a smooth frame-field with a feasible singularity structure on the tetrahedral mesh.
3) Computation of a seamless map on the tetrahedral mesh by constrained frame-field integration.
4) Quantization of the seamless map into an integer-grid map (IGM), enforcing alignment of critical points to integer map coordinates.
5) Hexahedral mesh extraction by pullback of the integer grid along the IGM.

Notably, all recent volumetric methods assume the singularity structure from **2)** to be fixed for the remaining steps – both spatially and combinatorially. This can pose a suboptimal restriction, as the early steps cannot properly take into account the discrete output mesh resolution determined in later steps, even though this aspect is strongly coupled to the singular structure. As seen in fig. 2, singularities of seamless and integer-grid maps are exactly those curves around which the total parametric angle differs from 2π (or π on the boundary), and therefore correspond exactly to chains of irregular edges in the later hexahedral mesh, with a valence different from 4 (or 2 on the boundary). An overly complex singular structure in the early stages will thus also result in a more irregular hexahedral mesh while meshes with sparse and well aligned irregular edges, so called *block-structured* meshes, are often preferred in applications.

In the surface setting, concerning quadrilateral mesh generation, allowing changes (in particular simplification) of the singular structure within the quantization step **4)** has already proven instrumental in achieving highest-quality results [LCK21b]. With the same goal in mind we propose a volumetric quantization method that incorporates the possibility of simplifying the singular structure, not as a heuristic preprocess or postprocess, but as integral part of step **4)**, steered directly by the minimization of a distortion measure that aims to preserve scale and spacing of mesh features between the continuous seamless map and the discrete integer-grid map.

1.1. Method Overview

Input Like similar recent quantization methods, our method takes as input a volumetric seamless map on a tetrahedral mesh domain. This input map implies an initial singular structure, defined by type, connectivity, and object space embedding of its singular curves. Additionally, we can respect marked 0D feature points, 1D feature curves and 2D feature surfaces, which elements of the output hexahedral mesh are required to exactly line up with. If no features are explicitly defined, we generally assume each domain boundary to be a feature surface, possibly partitioned into multiple such surfaces by singularities that lie within the domain boundary, treated as feature curves. The parametrization is required to be aligned with features and singularities.

Output Our method generates an integer-grid map (IGM), defined by the additional property that all critical points (singularities and features) lie on integer map coordinates. We explicitly allow the singular structure of the output IGM to differ from that of the input. More specifically, the resulting singular structure is never more complex than that of the input, but existing singularities are allowed to collapse or merge in various ways. The structure and object space positions of features are kept fixed, and a minimization of changes in feature scale, spacing, and directional alignment between input and output map is aimed for.

Procedure The transformation of the input seamless map to the output integer grid map proceeds in four substeps, each of which can be solved with hard robustness guarantees, as established in recent literature:

4a) Construction of a coarse T-mesh block decomposition of the input domain [BGMC22], each block being a parametrically-axis-aligned cuboid.

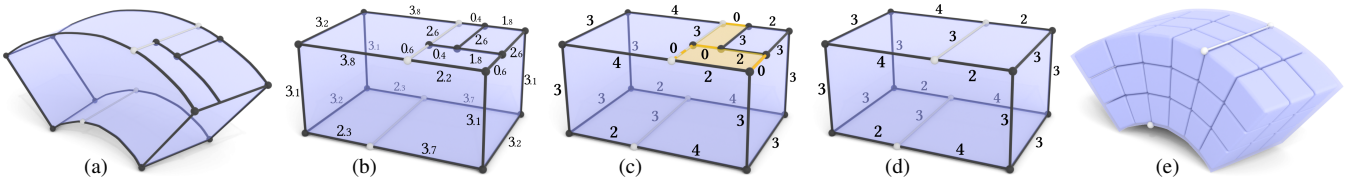


Figure 3: *The stages of our algorithm exemplified on a single T-mesh block: (a) A T-mesh is constructed on the input domain, possibly with features (white). (b) Under the input seamless map, each T-mesh element has a non-integer size. (c) Sizes are consistently quantized to non-negative integers, including 0 (orange). (d) Elements quantized to 0 are removed by collapsing, allowing an integer-grid map to be defined per block, matching the computed integers. (e) A hexahedral mesh is extracted as the preimage of the integer-grid, aligned with the features.*

4b) Global quantization, i.e. integer-assignment to the T-mesh which serves as a proxy for the underlying map [BBC22].
4c) Transformation of the T-mesh using collapse operators [BC23], removing all T-mesh elements quantized to an extent of 0.
4d) Reparametrization of each remaining T-mesh block, matching the assigned integer dimensions, forming an IGM, optionally followed by global map distortion minimization.

These steps are illustrated in fig. 3.

Our contribution lies within the central parts **4b)** and **4c)**. In the latter, we enable the moving, collapsing, and merging also of singular T-mesh elements, wherever demanded by the quantization. Most importantly though, we replace **4b)** by a novel quantization computation method. Firstly, in previous work the determination of constraints that guarantee structural integrity of the domain under quantized reparametrization – by preventing map degeneracies and holes or missing features in the output mesh – fundamentaly relies on an unalterable singular structure [BBC22, BC23, BBC24]. We introduce a new alternative constraint determination routine, that does not assume and require preservation of singularities, but still guarantees structure preservation of topological and user-defined geometric features, in section 4. Secondly, we provide a novel, higher-level volumetric quantization objective that promotes the large-scale preservation of features, their sizes and relative spacings, rather than focusing on local measures that do not actually carry over to the final output, as is prevalent in previous work. This goal is formulated as the minimization of a feature distortion measure defined on the T-mesh quantization variables, discussed in section 5. Background and notational conventions are established in section 3.

2. Related Work

The literature on algorithmic hexahedral mesh generation is vast [PCS*22]. We discuss here works with a particular relation to the various aspects of our proposed method.

Layout Subdivision The concept of quantization, fixing the integer degrees of freedom that determine the number of quads or hexahedra to pave each region of a domain with, has been the focus of sustained research also in the engineering and meshing community. An integer linear programming method was developed [TA93], assigning subdivision numbers to conforming patch and block layouts while minimizing the deviation of those from non-integer targets. A series of later works improved on this in terms of speed and balancedness, minimizing the deviations lexicographically [Mit00], or opting for heuristic [Mit14] or incremental [Mit23] approaches instead. These methods, however, assume that predetermined tidy conforming layouts are already given as input.

Integer-Grid Maps In the field of computer graphics, quantization for meshing was first approached in the context of mixed-integer parametrization. The domain is parametrized in a so called *integer-grid map* [BCE*13], such that the parametric integer-grid implies a quadrilateral or hexahedral mesh. As such, critical points of the map – more specifically singularities and features – must map to integer coordinates, see fig. 2. In loose relation to the above layout subdivision approach, the integer parametric spacings between critical points essentially represent the number of quads or hexahedra implied in between. Early works in both quad meshing [KNP07, BZK09] and hex meshing [NRP11] solved the mixed-integer parametrization problem via (sequential) rounding, This effectively favors speed and algorithmic simplicity at the cost of optimality and robustness, as interdependency of the integer variables is partially disregarded and map degeneration can be caused. The latter can be mitigated by suitable constraints [BCE*13], albeit at a high computational cost.

T-Meshes for Quantization A step forward in terms of both robustness and efficiency was the complete separation of integer and continuous degrees of freedom [CBK15]: First, a so called seamless map, the continuous relaxation of an integer-grid map (cf. fig. 2), is computed, and then used to construct a *motorcycle graph* partition [EGKT08] of the surface. This partition is a coarse, quadrilateral-only patch layout, often also referred to as a *T-mesh* partition due to being non-conforming, i.e. exhibiting T-junctions at patch boundaries (see fig. 4 for a volumetric T-mesh). Because all critical points of the input are also explicitly represented in the T-mesh, it can serve as a coarse proxy to determine valid and near-optimal integers for the underlying map [CBK15]. Like in layout subdivision, a quantization problem of assigning integer extents to all patches has to be solved, with the important differences of non-conformity and zeros explicitly being allowed. The latter is crucial, because the T-mesh might be arbitrarily untidy, in terms of containing very small or thin elements (see fig. 4). Allowing zeros introduces the need for explicit constraints preventing degeneration, i.e. guaranteeing separation of critical points. Once a valid proxy quantization has been determined, the integer-grid map can be computed using only continuous variables, with all integer degrees of freedom now predetermined [CBK15].

Building on this foundation, follow-up works have explored T-mesh collapse operators that remove 0-regions from the partition, enabling the construction of the integer-grid map in a simple and reliably injective patch-by-patch manner [LCBK19], or for generat-

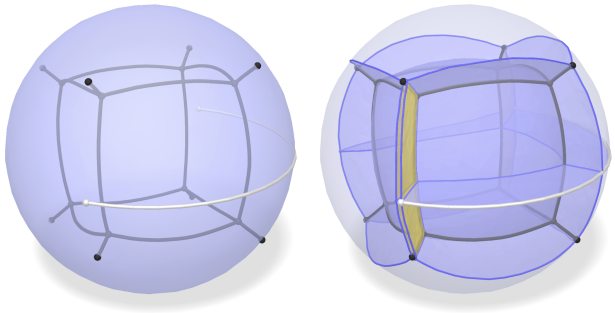


Figure 4: *Left: a parametrized volumetric domain with internal map singularities (black) and a surface feature curve (white). Right: a* T-mesh *constructed on the domain, i.e., a coarse, non-conforming cuboid decomposition that captures both features and singularities in its discrete elements, but may exhibit very slim regions (orange).*

ing coarse quadrilateral layouts [LCK21a, LCK21b], useful for the generation of neatly structured meshes, also called *block-structured*. Recently a coarse triangular partition has been suggested as a viable alternative to 2D T-meshes, serving in a similar manner as a proxy for efficient and reliable quantization of the underlying map [CODH*24]. For the general case of 3D volumetric quantization for hexahedral meshing, volumetric T-meshes termed *motorcycle complexes* [BGMC22] have been described as the first reliable alternative to the rounding approach only quite recently. They have thus far been used successfully for quantization [BBC22, BBC24] and collapse-based reliable reparametrization [BC23].

All of the above works adhere strictly to the predetermined singularity structure and embedding, usually stemming from previous frame-field computation – except for one method [LCK21b] (for the 2D surface case) that admits modifications to singularities in the form of limited translation and merging in pursuit of simpler quad layouts. In particular, this is shown to often be of great advantage for structure and element quality of the obtained quad mesh, due to the predetermined singularities often being suboptimal – generally, or specifically for the desired mesh resolution.

Frame-Field Singularity Modification Several methods address modifications to the singularity structure not as a part of quantization and for the sake of quality, but rather as a repair measure at an earlier stage to even just enable *meshability* of a frame-field, i.e. the existence of a corresponding seamless parametrization. On surfaces these measures usually only encompass the addition of a few singularities where necessary [MPZ14, PPM*16, PNA*21, SZC*22], while volumetric meshability is a much more complex issue requiring a whole suite of operators that e.g., merge, zip or unzip singular curves [FXBH16, JHW*14, LZC*18, RCR19, LB23]. In a more theoretical endeavor, it was demonstrated that any singular node, the branching point of singular curves, can be decomposed into non-branching singular curves [ZCFM23].

Hexahedral Mesh Singularity Modification More distantly related are works that modify the singularity structure after the fact, as a postprocess, on the hexahedral mesh. A typical goal is the explicit simplification of the mesh's intrinsic block layout, also called *base complex*. Such methods modify primal sheets of the hexahedral mesh [BBS02, LS10, GDC15], dual chords [TK03, KLSO12] or both, some also allowing singularities to be merged or split [GPW*17, XLZ*21, ZDLL25]. Operating in the realm of the already approximate discrete hexahedral mesh rather than in the realm of parametrizations of the original domain, poses challenges concerning the faithful preservation of boundary shape and features, amplified by these methods' often iterative nature.

3. Background

3.1. Volumetric Seamless Map

We assume our volumetric domain to be represented as a tetrahedral mesh $\mathcal{M}$; its elements will be referred to as tets, facets, edges, and vertices. The terms half-facets and half-edges will be used to denote oriented versions. A seamless parametrization on this domain is a chart-based map $\phi : \mathcal{M}_c \mapsto \mathbb{R}^3$, where $\mathcal{M}_c$ is the tetrahedral mesh (virtually) cut into a topological ball along some of its facets. The set of all such cut facets together is called the *cut surface*. The map ϕ is represented in a piecewise linear manner as per-vertex parametric coordinates $\mathbf{u} = (u, v, w)$, assuming vertices lying on the cut surface to be split into multiple (virtual) instances, one on each side of a cut.

The defining property of a seamless map is that vertex coordinates $\mathbf{u}$ and $\mathbf{u}'$ on front and back of any cut half-facet h are linked via a transition τ_h of the form [NRP11]:

$$\mathbf{u}' = \tau_h(\mathbf{u}) = R_h\mathbf{u} + d_h, \quad R_h \in \mathrm{Oct}, d_h \in \mathbb{R}^3, \tag{1}$$

where Oct is the rotational octahedral group, i.e., transitions permute the oriented parametric axes and translate. Singular edges of this map are exactly those edges for which the total dihedral parametric angle of incident tets differs from 2π (or π on the boundary). Due to the above, the difference δ is always an integer multiple of $\frac{\pi}{2}$ and the index of any singularity, identifying its type, is $i = \delta/2\pi$, i.e., an integer multiple of $\frac{1}{4}$. We call a maximal connected chain of singular edges without a branching point a singularity, while branching points between (and end points of) singularities are called singular nodes. Singularities must be aligned with a principal parametric axis. The defining property of integer-grid maps, a special case of seamless maps and the goal of our quantization, is the additional requirement that both non-aligned coordinates of singularities must be integers. All this is visually demonstrated in fig. 2.

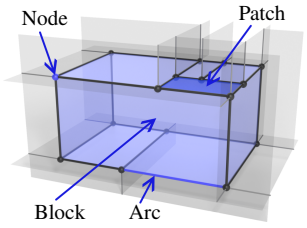
Critical Entities: Features and Singularities In previous work, the term critical entities was used to mean both singularities and user-defined features (points, curves, or surfaces). It was required that both should be aligned with the seamless map and should lie on integer coordinates of the later integer-grid map [BC23]. Additionally, it was demanded that neither may move (in object space), disappear or merge. For our work we refine this notion: We keep the map-related requirements for both types of critical entities – this part is necessary to obtain a feature-preserving hexahedral mesh – but we demand only features to not move, disappear, or merge; singularities are not prevented from doing so. Additionally, beside any user-defined features, we also explicitly preserve implicit (topological) features such as external boundaries, internal boundaries (voids) and handles (see section 4.3), whose preservation was previously implied by the preservation of singularities.

We assume the domain boundary facets to always be feature-marked and denote features by dimension in the following way:

0D *Feature nodes* are vertices marked as features.
1D *Feature curves* are maximal chains of edges marked as features, with equal alignment axis and uninterrupted by feature nodes or singularities. Feature curves must be either cyclic or bounded by feature nodes.
2D *Feature surfaces* are maximal area-connected sets of facets marked as features, with equal alignment axis and uninterrupted by feature arcs or singularities. Feature surfaces must be either closed or bounded by feature curves.

Notably, features are not confined to the boundary of the volume: Figure 13, for instance, demonstrates a feature surface embedded in the interior.

3.2. T-Mesh



T-meshes in our setting are a decomposition of the volume into cuboidal blocks, whose sides are axis-aligned with respect to the underlying parametrization. These blocks are bounded by 2D patches, patches are bounded by 1D arcs, and arcs are bounded by 0D nodes. Together these form a cell complex $\mathcal{T} = B \cup P \cup A \cup N$. Both object and parameter space geometry of this cell complex are defined by its embedding into the underlying tetrahedral mesh $\mathcal{M}$. Each element of $\mathcal{T}$ is embedded into a set of elements of corresponding dimension in $\mathcal{M}$. By construction, it can be ensured that all features and singularities of $\mathcal{M}$ also remain represented as discrete elements in $\mathcal{T}$. For the purpose of quantization, a T-Mesh can therefore serve as a coarse skeleton connecting globally all features and singularities, thereby allowing the coordinated assignment of quantized sizes and spacings.

3.3. T-Mesh Quantization

A T-mesh quantization is an integer vector $q = (q_1, ..., q_{|A|})$, assigning an integer q_i to each arc a_i, denoting its parametric length under the to-be-computed integer-grid map, thus implying the number of hexahedral sheets to be placed along that arc in the later hexahedral mesh. The total lengths of arcs on opposite sides of the same patch must match to guarantee the existence of a corresponding hexahedral subdivision for each block and thus for the entire domain. Integer parametric sizes and spacings should differ as little as possible from the non-integer sizes and spacings of the seamless input parametrization on $\mathcal{M}$. These continuous target sizes can be represented as a real-valued vector $\ell = (\ell_1, ..., \ell_{|A|})$ with per-arc target lengths ℓ_i, so that the objective becomes to minimize some distance $d(q, \ell)$.

As pathologically slim regions, discernable by $\ell_i < 0.5$, commonly occur in the T-mesh (see fig. 4), being able to locally assign $q_i = 0$ is crucial for the quality of the implied integer-grid map. Permitting zeros, however, introduces a risk: all features must remain represented and mutually separated in the later hexahedral mesh; hence, some regions need to be constrained to greater-than-zero size. In summary, a quantization q is valid *iff* it is:

- *consistent*, meaning that length sums of arcs on opposite patch sides must match,
- *structure-preserving*, meaning that the size of features as well as spacings in between must remain greater-than-zero, and
- *non-negative*, meaning that arcs (or blocks in a more general formulation) must not have negative lengths.

The existence of a valid quantization, in our setting based on a seamless-map-aligned T-mesh, is guaranteed; scaling the floating-point seamless lengths ℓ (representable as rational numbers, otherwise valid by construction) to integers yields a witness for this, as noted in previous 2D [CODH*24, CBK15] and 3D works [BBC22].

Integer program formulation Formally, the T-mesh quantization problem detailed above can be modeled as a linearly constrained integer problem with integer variable vector q [BBC24]:

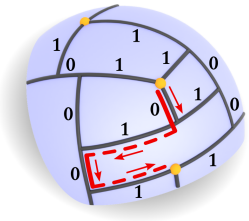
$$\begin{aligned} \min \quad & d(q, \ell) && && \text{(2a)} \\ \text{s.t.} \quad & \mathcal{A}q = \mathbf{0} && \text{(consistency)} && \text{(2b)} \\ & \mathcal{B}q \geqslant \mathbf{0} && \text{(non-negativity).} && \text{(2c)} \\ & \mathcal{C}q \geqslant \mathbf{1} && \text{(structure preservation)} && \text{(2d)} \end{aligned}$$

This problem can be solved by employing a general-purpose integer solver [BBC22] or efficiently approximated using a recent special-purpose solver [BBC24].

Structure preservation A quantization can fail to preserve the domain topology or features in the following ways:

Feature merge Two features (partially) merge under the quantization, implying a degeneration of the IGM in between (or enforcing a relocation of at least one of the features in object space).
Feature collapse A feature collapses to lower dimensionality under the quantization, e.g., a curve to a point or a region to a curve, implying a degeneration of the IGM along the feature (or enforcing a contraction of the features in object space).
Topology change The topology of the IGM changes, e.g., degeneration around a handle or around a cavity is implied (or their collapse in object space).

Due to the tight connection between topology of the domain and the singular structure of any boundary-aligned seamless map on it, in all but very specific edge cases (such as a hollow torus) the network of singular curves necessarily spans the topological features (handles, cavities) of the volume in such a way that simply maintaining the singular structure implicitly also preserves the domain topology without any additional care needed [BBC22]. Therefore, previous works that considered all singularities to be fixed, i.e. to-be-preserved features, only had to actively prevent collapse or merging of features in general. This can be done using simple constraints that ensure that the length of feature curves is $\geqslant 1$, together with additional feature *separation* constraints.



These separation constraints can be added lazily, upon detecting that one is violated. Detection of violation is essentially a graph search on the T-Mesh structure, starting from each critical entity and registering the shortest path that implies overlap with another critical entity under the quantization. This

path is a sequence of arcs; constraining a sum over these to be greater than 0 will enforce separation in the next solution iteration. If all singularities are considered features, all paths considered within the graph search are by construction always confined to regions of the seamless map that lie *between* singularities but never expand past these. Such regions are regular, so parametric directions of arcs are unambiguous. Therefore, notions like path segments being *parallel*, *antiparallel* or *perpendicular* have been used to determine signs for the sum constraints that ensure separation [BBC22]. Notably, this does not translate to our setting, where singularities are no longer fixed features and paths may extend beyond these and wind around these on their way to the nearest actual feature.

In section 4 we describe a novel method to generate structure-preserving constraints that separately and explicitly prevent changes to features *and* to domain topology, both without relying on a fixed singular structure. This requires tracing truly global paths within the T-mesh, i.e., ones that may pass by or wind around singularities. A axis-agnostic, sign-less approach is employed that makes extended use of the lazy constraint building framework. Within this framework, our approach may generate insufficiently strict constraints at first, but converges towards sufficient constraints through iteration, effectively consulting the quantization objective not only on what to separate but also on how to separate.

4. Structure Preservation with Flexible Singularities

In the following, we present a method to determine quantization constraints that together prevent all types of structure preservation failure listed above. As outlined in algorithm 1, a few of these constraints, $\mathcal{C}_0$, are formulated *a priori*, while for the remaining ones it is expedient to formulate and apply them only in a *lazy* manner. Repeatedly solving the problem with the current constraint set and adding new constraints proven to be violated by the previous solution, converges to a valid solution through iteration.

Algorithm 1: LAZY QUANTIZATION

Input: T-mesh with target lengths ℓ
Output: Final quantized integer arc lengths q
$q \leftarrow \mathbf{0}$
set up $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{C} \leftarrow \mathcal{C}_0$ // *a priori* structural constraints
repeat
  Minimize $d(q,\ell)$, s.t. $\mathcal{A}q = \mathbf{0}, \mathcal{B}q \geqslant \mathbf{0}, \mathcal{C}q \geqslant \mathbf{1}$
  $\mathcal{C}' \leftarrow$ LAZYSTRUCTURALCONSTRAINTS(q)
  if $\mathcal{C}'$ empty **then return** q
  $\mathcal{C} \leftarrow \begin{bmatrix} \mathcal{C} \\ \mathcal{C}' \end{bmatrix}$

The following subsections describe the novel construction of our more general and flexible constraints $\mathcal{C}$.

4.1. Preventing Feature Merge

An established way to lazily prevent the merging of features (see section 3.3) is to conduct a graph search on the T-mesh, find zero-paths between features that are witnesses for their merging, and constraining some arcs along those paths to become expanded in the next solution. With fixed singularities, this graph search can be directly based on the arc-node-graph of the T-mesh [BBC22]. The shape and size of features can be represented as 0D parametric points, or 1D and 2D parametric intervals. These intervals can be virtually carried along the path and checked for overlap. However, this does not work outside of parametrically regular regions, because transporting such an interval across a singularity implies ambiguous splitting or folding of the interval along that singularity (imagine, e.g., an interval being transported along the path in fig. 5).

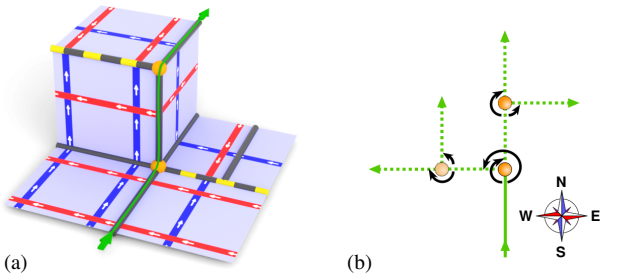
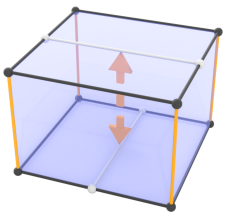


Figure 5: *(a) A T-mesh (black) on a surface domain with a seamless parametrization visualized as a texture, with singularities (orange) and cuts (yellow). An arc path (green) is indicated, crossing two singularities. (b) Depending on whether the path is interpreted as passing each singularity infinitesimally on their left or their right, the parametric directions of the path segments, relative to the starting segment, differ. This ambiguity is due to the transitions across cuts, or due to the parametric curvature at singularities.*



We therefore, instead of checking overlaps between arbitrary intervals under the quantization, consider only discrete points to simplify matters. Feature points alone however, are not sufficient to detect all types of feature conflicts, as demonstrated on the example on the right, where the orange arcs having 0-length implies merging of the two white feature arcs away from any node. Due to the integer-alignment of features, however, any two features that overlap parametrically, must overlap in at least one integer point. We therefore consider the integer points along all features to detect feature non-separation.

More concretely, we suggest building a graph whose nodes and connectivity are deduced from the current quantization integers, termed the *integer-grid graph*. We call the graph's nodes *integer-grid points*, which are the intersection points of any T-mesh element with the integer-grid implied by the current quantization integers, and its edges *links*, locally connecting any integer-grid points corresponding to the same vertex of the implied hexahedral mesh.

4.1.1. Integer-grid graph

We begin with the detailed construction routine for the integer-grid graph. W.l.o.g. we assume the interior of blocks to be transition-free, with non-trivial transitions moved onto patches of the T-mesh. Then, after fixing an arbitrary node per block as the origin $(0,0,0)$, block-local integer-grid coordinates per node n can be deduced from the quantized length and direction of that block's arcs. In each block, this yields a single integer-grid point per contained node n.

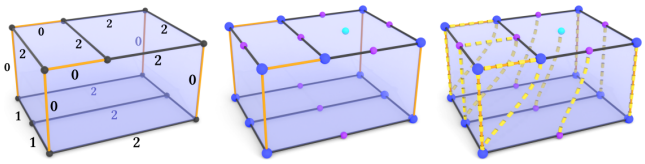


Figure 6: *On a T-mesh with preliminary quantization integers (left), the integer grid graph is constructed by forming integer-grid points on each T-mesh element for each implied intersection with the integer grid (center), and connecting those by links that correspond to the same integer coordinates (right).*

Contained arcs a cover integer-grid intervals of length q_a. To avoid redundancy, only integer-grid points in their interior are modelled, yielding $\max(0, q_a - 1)$ integer-grid points per arc, interpolated between the two bounding node coordinates. The integer-grid rectangle formed by patches p with quantized side lengths q_p^w and q_p^h then covers $\max(0, (q_p^w - 1)(q_p^h - 1))$ interior grid points, obtained by interpolation between the four patch corner nodes. Links are (implicitly) created within each block between any two integer-grid points sharing the same local integer-grid coordinates.

The construction process per block is summarized in fig. 6. Across blocks, the transitions between blocks define a correspondence between the per-block integer-grid points and thereby connect the block-wise graphs into a global one.

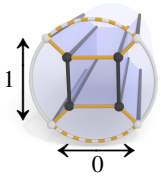
4.1.2. Searching the integer-grid graph

By searching this graph, starting from an integer-grid point on a feature, we can quickly determine all other points on features that would collapse under the quantization. Not all paths collapsing two different integer-grid points on features are problematic, though. Those that do not require separation specifically include:

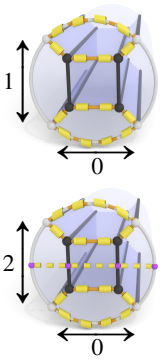
1. Paths entirely within one feature.
2. Paths entirely within the union of start and end feature, i.e. paths between features that are adjacent via a shared boundary (or paths homotopic to those). Separating these features is neither necessary nor feasible, as they are connected already in the input.

Both the above can be excluded by first conquering all integer-grid points reachable by paths that never leave the starting feature or an adjacent feature, and marking these points as unproblematic. Then the actual search can be conducted, creating a separation constraint for any problematic integer-grid point encountered from the starting point. One full lazy separation iteration then repeats the above, starting a search from each feature integer-grid point and adding any new separation constraints determined in these searches to the current quantization problem.

Non-simple features By definition, feature curves are straight and feature surfaces planar with respect to the underlying parametrization. Through transitions, they can still be topologically more complex than a segment or a disk, respectively, called *non-simple* here for short. Within non-simple features, like the cyclic feature curve (white) in the inset, it is possible that two opposite non-zero regions exist (left and right arc in the inset), whose boundaries (white nodes) are implied to collapse onto each other *within* the containing feature, due to being part of the same zero-regions. These collapses, due to being executable within the feature (orange-white in the inset), are correctly considered unproblematic by the above routine.



However, in the case that the two non-zero regions have no integer-grid points in their interior (top in the inset), marking their merging boundaries as unproblematic inadvertently masks the potential collapse of their interiors onto each other. This interior collapse should be considered problematic, due to being executable only *outside* the feature. Fortunately, the solution within the framework of the integer-grid graph is simple: Before starting searches from non-simple features, virtually scale up the quantization integers by factor 2; this guarantees that an integer-grid point is present on the interior of any non-zero arc and patch (bottom in the inset). From these virtual interior integer-grid points a problematic collapse of arcs or patches onto each other is correctly detected, as their collapse path does not lie within the unproblematic zero-regions.



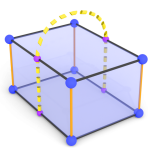
4.1.3. Search optimization

By performing the graph search in a Dijkstra shortest-path manner, and skipping sectors that problematic points have already been found in, as suggested in [BBC22], we can limit the lazy separation constraint construction to *nearest* neighbors per iteration, heavily reducing the runtime per iteration while increasing the iteration count only slightly.

The integer-grid graph has a size that scales with the output hex mesh size. We can reduce this to scale only with the T-mesh size via the following *bundling* formalism. Recall that arcs are axis-aligned intervals and patches are rectangles under the quantization. The integer-grid points in their interior are exactly those covered by the (closed) arc-interval or patch-rectangle shrunk from all sides by 1. Thus, integer-grid points can be bundled per containing arc or patch into such 1D or 2D intervals (represented via only two extremal integer-grid points). Links can likewise be bundled: The bundle of links between two distinct point bundles corresponds to their geometric intersection. Hence, the integer-grid graph can be traversed via intersection queries between intervals, without explicitly constructing individual points and links.

4.1.4. Constraint formulation

Once two points on two different features and a separation-worthy path in between, made of links in the integer-grid graph, have been identified, a constraint must be formulated based on T-mesh arcs. To convert the link path into a sequence of arcs, we determine the T-nodes parametrically closest to the start and end points of the link-path. Then the parametrically shortest arc-path between start and end nodes is computed by a Dijkstra search, confined to arcs of the blocks traversed by the link path. In the rarely occurring case that a link path crosses a block more than once, we instead apply the above procedure to each link of the path individually, and stitch the resulting arc paths to



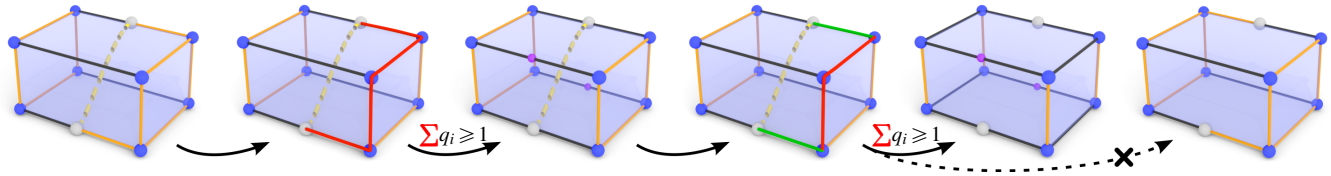


Figure 7: *The lazy constraint building process repeatedly finds separation-worthy paths (yellow) between features (white) on the integer-grid graph, translates these graph paths into arc paths in the T-mesh (red/green), and constrains the length sum over all currently zero-length arcs in the path (red) to be positive. Repeating this eventually leads to separation of the features. Note that the constraints can be conservative; the quantization on the far right, while also a valid alternative, is no longer within the feasible set of the constructed separation constraints.*

build the whole path. This way we ensure that the arc path is always homotopic to the link path.

We then constrain the sum over the quantized lengths of all 0-arcs (arcs with currently $q_i = 0$) in the path to be $\geqslant 1$. The idea behind this sum-based way of formulating the constraint is to effectively let the solver choose which arcs are least costly to "inflate" in terms of the quantization (and even freely update this in subsequent iterations), instead of us making a pick. A key difference in this constraint formulation to that of [BBC22] is that, due to the direction ambiguity in our flexible-singularity setting as illustrated in fig. 5, we use a completely direction agnostic approach: the relative parametric direction of arcs in the paths does not play a role in the above constraint formulation. This is in contrast to the *directed signed* constraints used in previous work, relying on the notion of *opposite* and *transversal* orientation within the path.

As a consequence of this direction agnostic formulation, such a constraint does not always separate the two features right away, see fig. 7 for an example. However, subsequent iterations of lazy constraint addition are guaranteed to take care of this eventually, as follows from the below argument:

- If the current solution is infeasible (i.e., not structure-preserving), a witness for this is found, in the form of a node-arc path, consisting of arcs of which at least one is a zero-arc.
- A constraint over the zero-arcs in this path is added, forcing this arc subset to attain a quantized length sum >0 in the next iteration. Because the constraint is violated by the current solution, the constrained arc subset is new (previously unconstrained).
- Hence, the set of constrained arc subsets grows in each iteration. Because the set of arc subsets is finite, and (in the extreme case) constraining all arc subsets yields a trivially feasible solution without zero-arcs, a feasible solution is found by finite iteration.

4.2. Preventing Feature Collapse

Preventing the collapse of features to lower dimensionality is applicable to 1D feature curves, represented in the T-mesh by a chain of arcs, and 2D feature surfaces, represented by a collection of neighboring patches. Preserving feature curves is easy using *a priori* constraints: we simply demand for each feature curve that the sum of quantized arc lengths along the feature curve is positive.

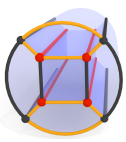
Preserving feature surfaces is not as straightforward, because, e.g., demanding a positive area in a direct way would require non-linear constraints. However, as feature surfaces are bounded by feature curves by definition, and these are already constrained to length $\geqslant 1$ each, many feature surfaces do not require dedicated attention. In particular, as soon as on a disk-topology feature surface boundary there are two adjacent feature curves with different parametric axis alignment, a non-zero parametric area of the feature surface is already implied by the curves' constraints. This leaves, e.g., closed feature surfaces and ones with cyclic boundary feature curves to be handled. We will show that these do not require separate attention either, because the below rigorous topology preservation constraints implicitly also preserve these feature surfaces.

4.3. Preventing Topology Change

A quantization length of 0 assigned to an arc or arc path effectively means a contraction of its two end points onto one another in parameter space. If this were to happen to an (arc) loop that is topologically *incontractible* (in $\mathcal{M}$ or in $\partial\mathcal{M}$), a topological mismatch between object and parametrization would be implied, as object and T-mesh (after collapsing 0-elements) were no longer homeomorphic. We therefore derive quantization constraints based on such loops.

Loops incontractible in $\mathcal{M}$ are captured by the fundamental group $\pi_1(\mathcal{M})$ of our domain $\mathcal{M}$. More precisely, to conveniently include possible volume-internal feature surfaces right into this consideration, we consider $\pi_1(\mathcal{M}')$, where $\mathcal{M}' = \mathcal{M}\backslash S$ and S is the set of feature surfaces. In section 4.3.1 we describe how to form constraints that preserve this fundamental group by preventing the quantization from collapsing a selected set of loops that generate this group.

Loops incontractible in the boundary $\partial\mathcal{M}'$ are captured by the fundamental group $\pi_1(\partial\mathcal{M}')$ of the domain boundary. The nontrivial loops that generate this group are of two types: those that are incontractible in $\mathcal{M}'$ (considered above already, thus requiring no further attention) and those that, while incontractible in the boundary, are contractible in $\mathcal{M}'$. The latter loops still need our attention. Due to being contractible in $\mathcal{M}'$, they bound disk-topology surfaces (cross sections) in $\mathcal{M}'$. By the Gauss-Bonnet theorem, the seamless map's restriction to such a cross section must, due to its disk-topology, contain singularities of total index -1. This implies that these loops do not require further attention either – due to the singularity index bounds we are going to impose in section 4.3.2. Concretely, we prevent singularities from merging in such a way that indices $< -\frac{1}{4}$ are formed. Hence, multiple singular curves remain separated under the quantization, as illustrated in red in the inset cross section example, precluding a collapse of the encompassing loop. Note that even the indicated quantization with



orange 0-arcs, which would not fully collapse the loop to a point but to a segment, is precluded in this way.

Finally, the second homotopy groups, $\pi_2(\mathcal{M}')$ and $\pi_2(\partial\mathcal{M}')$, capture *incompressible* spheres that, analogous to incontractible loops, must be prevented from degenerating. Higher homotopy groups play no role due to our three-dimensional setting. $\pi_2(\partial\mathcal{M}')$ is non-empty for sphere-topology boundary components. The restriction of the seamless map to such a sphere-topology boundary surface, however, necessarily contains negative index singularities (by the Gauss-Bonnet theorem), so our lower singularity index bound of $-\frac{1}{4}$ already prevents it from collapsing by separating multiple singularities in touch with the surface. If $\pi_2(\mathcal{M}')$ is non-empty, there are spheres that are incompressible within $\mathcal{M}'$, implying the presence of cutouts in the interior. Because of feature surfaces, such cutouts might even be nested. Note, however, that such nested features are already separated mutually and from the outer boundary via the separation constraints established in section 4.1. Hence, we need only consider each innermost surface $\partial\mathcal{M}'_i$. It turns out that no further dedicated treatment is necessary: if $\partial\mathcal{M}'_i$ is of sphere-topology, the above index argument applies; otherwise, there are incontractible loops in $\partial\mathcal{M}'_i$ and these are either contractible in $\mathcal{M}'$ and thus already handled by the singularity index bound (cross section argument), or incontractible in $\mathcal{M}'$ and thus already handled via $\pi_1(\mathcal{M}')$-constraints.

4.3.1. Preservation of $\pi_1(\mathcal{M}')$

Quantizations implying the contraction of loops that are incontractible in $\mathcal{M}'$ need to be prevented. The idea is to detect when a connected component in the integer-grid graph, i.e. a zero-region, is not simply connected. This indicates zero-loops wrapping around topological features (violations of incontractibility) or around certain clusters of negative-index singularities (violations of index bounds). Explicit distinction is not necessary as both must be prevented; the latter is also (more generally) handled in section 4.3.2. Other types of regions cannot be wrapped by a zero-region as consistency constraints would transitively imply them being part of the zero-region.

A non-simply connected zero-region in the integer-grid graph can be detected by the following procedure, akin to a tree-cotree decomposition [Epp03]. First, compute a spanning tree of the zero-region. Let the co-tree contain all untraversed links of the region. Each of these links, when inserted into the tree, would form a loop – contractible (within the zero-region) or incontractible. To exclude the contractible ones, we remove links, one by one, from the co-tree (inserting them into the former tree) if they close any loop that is entirely contained within one T-mesh block – which trivially implies its contractibility. This is iterated so as to, transitively, also exclude links closing larger contractible loops. Afterwards, for each remaining link in the co-tree, for the root-based loop it closes, a constraint is formulated as described in section 4.1.4, eventually enforcing the inflation of this loop in the quantization.

This does not need to be done everywhere. Instead, consider a *cut surface* that virtually cuts $\mathcal{M}'$ into a topological ball. Any incontractible loop must pass through this cut surface, so searching only from integer-grid points on this cut surface is sufficient. To compute a discrete cut surface $C \subset P$ we initialize C with all patches P, perform a breadth-first search on the dual of the T-mesh $\mathcal{T}'$, removing crossed patches from C. Then "dead-end" patches, containing any arc incident to no other *cut* patch, are removed from C iteratively, and finally boundary patches are removed.

4.3.2. Singularity index bounding

As we do not treat the input singularity structure as fixed, we need to take care that only permissible modifications occur. When singular curves merge (due to not being separated by the quantization) the resulting curve has an index that is the sum of the original indices. We wish to exclude mergings that lead to combined indices out of a range $[I_{\text{low}}, I_{\text{high}}]$ of acceptable indices. Most importantly, we set $I_{\text{low}} = -\frac{1}{4}$ as singularities of lower index would induce hex mesh edges of valence $\leqslant 2$ and non-convex hexahedra (with inner angles $\geqslant 180^\circ$) that are at least unfavorable, if not unsuitable in practical use cases. I_{high} is an optional, adjustable upper index bound.

We employ a variation of the integer-grid graph (section 4.1.1) for detection and separation of clusters of singularities merged by the current quantization that have out-of-bounds indices. As opposed to the previous considerations on feature separation, we do not care for vertex-based coincidences, as two singular curves joining merely at a common node are not relevant here. Instead, we care about singular curves overlapping along a 1D segment. The shortest possible such segment is one edge of the implied hexahedral mesh. So the integer-grid graph must be adapted to be based on implied edges, rather than vertices. We observe that there is a bijection between the set of implied hexahedral edges and points that are half-integers in a single coordinate and integers in the remaining two coordinates. So, if we consider a quantization of exactly twice the resolution per axis, any integer-grid point with an odd integer in one coordinate and even integers in the other two represents a hexahedral edge.

On the double-resolution integer-grid graph filtered to specifically these points, we start a search rooted at an edge-representative integer-grid point on a singularity. The resulting search tree, made from all links on shortest paths starting from the root, is pruned to a spanning tree of all the reached singular integer-grid points. We call the set of these singular integer-grid points a *singular cluster*, and each chain of integer-grid graph links connecting two of such points within the spanning tree a *cluster chain*. Any singularities within the singular cluster would merge under the quantization, leaving a single singular (or regular) edge with an index equal to the cluster's index sum. That index sum I is determined and compared with the index bounds $[I_{\text{low}}, I_{\text{high}}]$. If I is within bounds, we leave it as is. Otherwise, the index distance ΔI to the closest bound is computed. For each cluster chain, we compute the index sums I_1 and I_2 of the two sub-clusters it connects. Splitting any chain for which $\max(\Delta I_1, \Delta I_2) < \Delta I$ brings the worst index closer to within the bounds; repetition results in sub-clusters with only inside-bounds indices.

Here again, we leave the choice of which link to split to the solver, to the outer objective minimization, instead of making a narrowing, heuristic choice. To do this, for each chain where $\max(\Delta I_1, \Delta I_2) < \Delta I$, we transform its link path into an arc path as discussed in section 4.1.4, and then simply formulate a constraint that demands the sum over all lengths of 0-arcs in those paths to be $\geqslant 1$. Repeating this will eventually dissolve or recombine all out-of-bounds singularity clusters, leaving only valid ones. Note

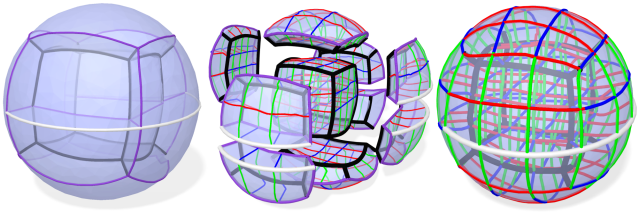


Figure 8: *On the T-mesh of a domain (left), the integer-grid map can be initialized with non-degeneracy guarantees on a block-wise level (center), and then be globally optimized for smoothness (right); only features (white) and singularities (black) remain fixed.*

that within the above, we have assumed that the input singularities meet the index bounds. In case the input contains out-of-bounds singularities, they either improve through objective-driven merging, or remain unchanged, but will never worsen.

5. Flexible Quantization Objective

The goal of the quantization's objective function is to minimize the implied deviation of the output integer-grid map from the input seamless map. Previous works on volumetric quantization used an objective function that punishes the deviation of each individual arc's length from its target. Specifically eq. (2a) becomes

$$d_{\text{arc}}(q,\ell) = \sum_{a_i \in \text{arcs}} (q_i - \ell_i)^2 = (q-\ell)^\mathsf{T} \mathcal{I} (q-\ell) = ||q-\ell||^2_{\mathcal{I}}, \quad (3)$$

where $\mathcal{I}$ is the identity matrix. However, the lengths of individual arcs should not actually matter to us – they do not carry over to the final IGM or hexahedral mesh. They may be relevant for a block-wise initialization of the IGM, but become irrelevant once a global smoothing optimization is applied, as illustrated in fig. 8. Only the *accumulated* integer spacings between and along features matter, as these features are fundamentally fixed in object space.

For the goal of maintaining feature sizing and spacing, the arc-based objective can even be strongly detrimental. Imagine two features separated by a straight chain of N arcs, each with target length $\ell_i = 0.4$. Under the above objective, assigning an integer length of 0 to each, and thus a total length of 0 to the chain, is considered optimal. Instead, the objective should only consider the relative spacings between features, as only these are invariant in the later steps of the meshing pipeline, and encourage the chain of arcs to have a total length of as close to $0.4N$ as possible in this example. In particular, the result should not be strongly influenced by the density of arcs in the domain, which is related to the density of singularities.

To generalize this intuition, we define in the following a novel *feature-based* rather than *arc-based* (and thereby indirectly also singularity-based) objective, focused on paths between features; it is independent of the density of the T-mesh arc sampling induced by the singularity structure and thus serves as a high-level decision maker for the overall goal of adaptively simplifying the singular structure wherever beneficial. To maintain the benefits and simplicity of a quadratic objective, our objective will be of similar form,

$$d_{\text{feature}}(q,\ell) = (q-\ell)^\mathsf{T} \mathcal{F} (q-\ell) = ||q-\ell||^2_{\mathcal{F}}. \quad (4)$$

The matrix $\mathcal{F}$ will be responsible for accumulating individual arcs into feature-spanning and feature-connecting paths, as well as weighting these appropriately. As a conceptual basis, we will first introduce a general measure of feature distortion between integer-grid map and seamless map and then show how that measure is efficiently approximated on top of our T-mesh.

5.1. Feature Distortion Measure

When comparing seamless and quantized integer-grid map, represented as target and quantized T-mesh arc lengths, our overall goal is to measure and minimize the distortion of size and relative position of features. Relative positions of features can be assessed based on paths between these. Due to transitivity, it is sufficient to consider paths between neighboring features instead of paths between all pairs. The notion of neighborhood should be as tight as possible to avoid redundancy, but sufficiently broad to not leave gaps in the resulting path network. As a natural neighborhood notion, we suggest the use of Voronoi adjacency, i.e. two features are considered neighboring if their cells in a volumetric Voronoi diagram (in the metric of the seamless map) are adjacent. Based on the resulting path network we can measure feature distortion, and formulate a volume integral of this measure to make it agnostic to network density. Both path length and direction are important to preserve, because spacing per axis matters. Concretely, let E be the set of paths, $\vec{e}_\ell$ and $\vec{e}_q$ the vector from start to end of $e \in E$ in (a local chart of) the seamless map and quantized map, respectively, and $V_\ell(e)$ some part of the volume (in the seamless map metric) exclusively assigned to it, the suggested relative distortion measure to minimize then is

$$d_{\text{feature}}(q,\ell) = \sum_{e \in E} V_\ell(e) \frac{||\vec{e}_q - \vec{e}_\ell||^2}{||\vec{e}_\ell||^2}. \quad (5)$$

In the following we clarify how to express $V_\ell(e)$, $\vec{e}_\ell$ and $\vec{e}_q$ in terms of arc-based targets ℓ and quantization variables q.

We remark that the path network derived from Voronoi adjacency is akin to a conforming Delaunay tetrahedralization of the features – with the caveat that the latter is not entirely well-defined in the metric of the seamless map due to the singularities. Interestingly, the utility of a Delaunay tessellation for quantization was pointed out before, for the 2D case [CODH*24]. Importantly though, in that work it was used differently, as an alternative to the T-mesh, limiting it to the 2D surface setting. Here we instead employ it as a guiding principle on top of the T-mesh, as detailed next.

5.2. Approximation on the T-Mesh

We construct a set E of paths, guided by the above concept, as chains of arcs by computing approximate Voronoi cells directly on the basis of the discrete T-mesh structure.

Voronoi diagram As seeds for the Voronoi cells we use feature nodes. As feature curves are bounded by such nodes, and feature surfaces are bounded by feature curves bounded by such nodes, in this way the resulting paths conveniently capture both inter-feature spacing as well as intra-feature sizing. To properly include node-less and node-poor features in this, on feature curves with $\leqslant 2$ feature nodes, i.e. cyclic ones, and on feature surfaces with $\leqslant 4$ feature nodes, we greedily add maximally distant pseudo feature nodes.

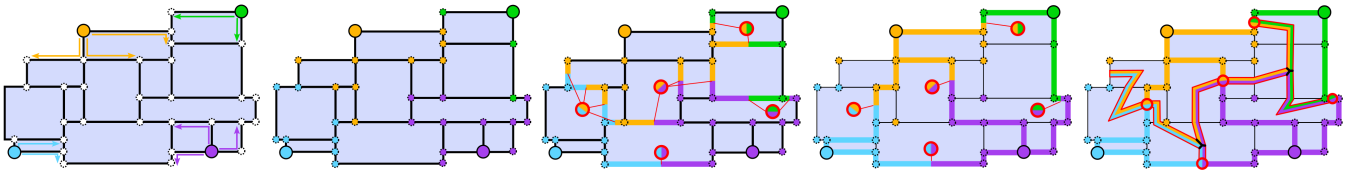
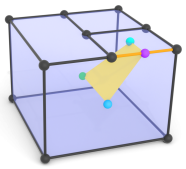


Figure 9: *The T-mesh based approximation of a feature-based Voronoi diagram of the parameter space and resulting Delaunay edges between features, illustrated on a 2D example. (a) Starting from feature nodes, Voronoi regions are grown via arc paths. (b) Regions are formed by all nodes of the same coloring. (c) Interfaces between regions are arcs with different color pairings at their end nodes. (d) The shortest path through each interface arc group is determined. (e) The interface area is estimated based on the (barycentric) dual of interface arcs.*

From these (pseudo) feature nodes, concurrent Dijkstra shortest path searches on the node-arc-graph of the T-mesh are started, using ℓ as arc weights. T-mesh nodes are colored according to the seed point they were first reached from, see fig. 9. Voronoi cells are formed by (dual cells of) nodes of the same color. Each arc connecting two nodes of different color is marked as an interface arc. Interface arcs are then grouped into *interfaces*, connected components (where arcs are considered connected if adjacent to a common patch) with constant color pairing. Each such interface corresponds to a face of the discrete volumetric Voronoi diagram, shared by two cells. Lastly, to form the set of arc-paths E, the shortest arc-path through each such interface connecting its two seeds is determined.

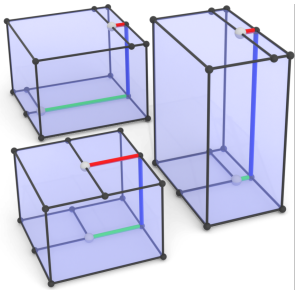
Volume integral The seamless map volume $V_\ell(e)$ pertaining to each such path $e \in E$ can be conveniently estimated as $V_\ell(e) = \frac{1}{3} A_\ell(e)\,||\vec{e}_\ell||$, where $A_\ell(e)$ is the area of the interface traversed by e, i.e. the sum of areas of dual faces of the interface's arcs in the T-mesh's barycentric dual. This is computed per interface arc a as follows: For each incident block b_a compute the parametric dual area attributed to a, by triangulation of the quad spanned by the midpoint of a (purple), the midpoints of the two patches incident on both a and b_a (cyan), and the midpoint of b_a (green).



The idea behind this is an imagined partitioning of a Voronoi cell into (polygonal) cones, apex at the seed, with the interfaces as bases, attributing a cone's volume to the corresponding interface's path.

Path vectors The vectors $\vec{e}_\ell$ and $\vec{e}_q$ of an arc path are not uniquely defined, in particular when crossing singularities. The 3D parametric difference vector between start and end node is ambiguous due to the lack of a global regular coordinate system, as in fig. 5.

We could circumvent this by just using the sum of the path's arcs' lengths, ignoring relative orientation. But this would approximate the vector length only crudely and would ignore directional information entirely. The inset demonstrates three different quantizations of the same T-mesh block. Despite the two white feature nodes being in very different spatial relation, the sum of arc lengths along the red-blue-green arc path is the same. To actually discern these, the information that the blue arc lies along a different axis than red and green, and the information that red and green arcs are in opposite orientation along the path, is important.



We heuristically disambiguate the relative directional ambiguity when an arc-path from E crosses a singularity, by assuming it infinitesimally passes by the singularity, and of the multiple ways there are to pass a singularity (two in the case of singular curves, more in the case of singular nodes) picking a shortest one, i.e. smallest turning angles. For this purpose, lengths and angles are measured with respect to the input seamless map, represented by arc lengths ℓ and (per-block) arc orientations. The rationale behind this is that the arc-paths in E are (discrete approximations of) parametrically shortest paths. In this way axis direction and orientation labels for all arcs along a path, relative to the path's first arc, are determined. We express the resulting axis and orientation information per arc a_i in an arc-path e by defining for each triplet (e, a_i, dir), with $dir \in \{\vec{u}, \vec{v}, \vec{w}\}$, a ternary sign:

$$s_{e,i}^{dir} = \begin{cases} 1 & \text{if within } e \;\; a_i \text{ is traversed along } dir \text{ in pos. direction} \\ -1 & \text{"} \qquad\quad\;\; a_i \text{ is traversed along } dir \text{ in neg. direction} \\ 0 & \text{"} \qquad\quad\;\; a_i \text{ is not traversed along } dir \end{cases}$$

Using this we can now express the edge vector difference in terms of arc variables q_i:

$$||\vec{e}_q - \vec{e}_\ell||^2 = \sum_{dir \in \{\vec{u},\vec{v},\vec{w}\}} \left(\sum_{a_i \in A} s_{e,i}^{dir} (q_i - \ell_i) \right)^2 .$$

Final objective Using the above, (5) is expressed in arc variables:

$$\begin{aligned} d_{\text{feature}}(q,\ell) &= \sum_{e \in E} w_e \sum_{dir} \left(\sum_i s_{e,i}^{dir} (q_i - \ell_i) \right)^2 , \\ w_e &= \frac{V_\ell(e)}{||\vec{e}_\ell||^2} = \frac{A_\ell(e)}{3\,||\vec{e}_\ell||} = \frac{A_\ell(e)}{3\sqrt{\sum_{dir} \left(\sum_i s_{e,i}^{dir} \ell_i \right)^2}} . \end{aligned} \tag{7}$$

It can be rearranged into matrix form

$$d_{\text{feature}}(q,\ell) = (q-\ell)^\mathsf{T} \mathcal{S}^\mathsf{T} \mathcal{W} \mathcal{S} (q-\ell) = (q-\ell)^\mathsf{T} \mathcal{F} (q-\ell) , \tag{8}$$

where the matrices $\mathcal{S}$ and $\mathcal{W}$ are effectively responsible for accumulating arcs into arc paths and weighting the paths, respectively:

$$\mathcal{S} = \begin{pmatrix} s^{\vec{u}}_{1,1} & s^{\vec{u}}_{1,2} & \cdots & s^{\vec{u}}_{1,|A|} \\ s^{\vec{v}}_{1,1} & s^{\vec{v}}_{1,2} & \cdots & s^{\vec{v}}_{1,|A|} \\ s^{\vec{w}}_{1,1} & s^{\vec{w}}_{1,2} & \cdots & s^{\vec{w}}_{1,|A|} \\ s^{\vec{u}}_{2,1} & s^{\vec{u}}_{2,2} & \cdots & s^{\vec{u}}_{2,|A|} \\ \vdots & \vdots & \ddots & \vdots \\ s^{\vec{w}}_{|E|,1} & s^{\vec{w}}_{|E|,2} & \cdots & s^{\vec{w}}_{|E|,|A|} \end{pmatrix},$$

$$\mathcal{W} = \operatorname{diag}\left(w_1, w_1, w_1, w_2, w_2, w_2, \ldots, w_{|E|}, w_{|E|}, w_{|E|}\right).$$

Note that each arc path in E often extends along only one or two axes, so $\mathcal{S}$ may contain many zero-rows; these rows and the corresponding rows and columns in $\mathcal{W}$ can be dropped. Furthermore, matrix $\mathcal{F}$ commonly is not full-rank as not all arcs are covered by E. It is therefore sensible to use the arc-based objective (3) as a regularizer,

$$\begin{aligned} d_{\text{total}}(q,\ell) &= d_{\text{feature}}(q,\ell) + \lambda d_{\text{arc}}(q,\ell) \\ &= ||q-\ell||^2_{\mathcal{F}+\lambda\mathcal{I}}, \end{aligned} \tag{10}$$

with λ chosen adequately small (e.g. $10^{-3}V_\ell$). This avoids numerical problems in common optimization schemes and favors solutions with lower arc stretch from among those with equal distortion.

6. Results

We have implemented our method based on two open source libraries of previous work [BBC22, BC23] and integrated it into these. They are available at www.github.com/HendrikBrueckler/QGP3D and www.github.com/HendrikBrueckler/C4HexMeshing.

6.1. Experiments

We apply our method to those datasets of seamless parametrizations employed for testing also in former recent publications on volumetric quantization [BBC22, BC23, BBC24] and compare our results to theirs. Out of the 197 total models, 103 come with feature markers and 60 additional ones contain sharp boundary singularities that we treat as feature curves, yielding 163 models with multiple feature types and 34 models without features (except for their boundary, implicitly considered a feature surface). To cover a wide range of scenarios, for each of these models we target seven different output resolutions, by creating seven instances per model with the seamless map isotropically scaled to a parametric volume of $V_\ell \in \{10^0, 10^1, ..., 10^6\}$. Note that this volume V_ℓ indicates the target number of hexahedra to roughly be expected in the output. Previous methods have used a target volume of 0 when aiming for the coarsest quantization, effectively setting all target lengths to 0 [BBC22, BC23]. We use a lowest setting of 1 instead, which sets all arc target lengths to very small yet distinct values, maintaining a preference among them – and allowing safe computation of relative length distortions within our novel objective via eq. (7). This setting is used across all compared methods for fairness.

For each of the 1372 instances, in order to thoroughly analyze our method's ingredients, we compute quantizations in each of the following five ways:

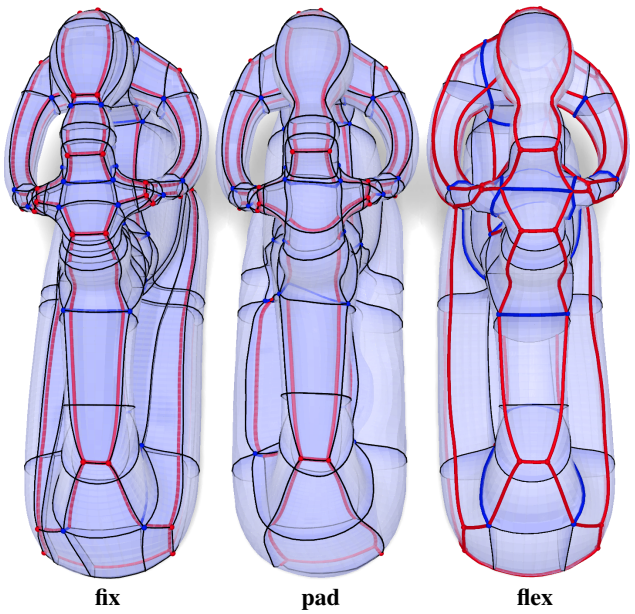


Figure 10: *Base complex induced by three different quantization modes for handling singularities: Fixed (left), flexible singularities except that inner singularities must not merge into the boundary (center), and completely flexible (right). Valence 3 and 5 singularities in red and blue, respectively. The center option is interesting for applications that may suffer from singular curves on the boundary.*

$q_{\textbf{arc,fix}}$ Minimize the arc-based objective with fixed singularities (as in [BBC22, BC23]).

$q_{\textbf{feat,flex}}$ Minimize the feature-based objective with flexible singularities (our novel method).

$q_{\textbf{feat,pad}}$ Minimize the feature-based objective with flexible singularities, but keep inner singularities inside (our novel method with the option of *padding-preservation*).

$q_{\textbf{feat,fix}}$ Minimize the feature-based objective with fixed singularities (ablation scenario I).

$q_{\textbf{arc,flex}}$ Minimize the arc-based objective with flexible singularities (ablation scenario II).

Figure 10 demonstrates the effect of padding-preservation. As index bounds for singularities we use $[-\frac{1}{4}, \frac{2}{4}]$. From the quantizations hexahedral meshes are generated using the pipeline detailed below.

6.2. Complete Pipeline

T-mesh construction T-meshes are constructed on the input map using the *Motorcycle Complex* method [BGMC22], including the refinement option that prevents self-incidences of T-mesh elements.

T-mesh quantization A quantization is computed as summarized in algorithm 1, with constraints and objective formulated as described in sections 4 and 5, respectively. As the solver for the central integer quadratic program, we use the general-purpose mixed-integer solver GUROBI or alternatively the problem-specific *integer-sheet-pump* solver ISP [BBC24] – which we adjust for more effective use with

our novel objective as detailed in Appendix A. GUROBI was configured with a time limit of 10 minutes per solve, returning the best available solution at that cut-off.

T-mesh collapsing Any T-mesh elements quantized to a size of 0 are collapsed using generalized collapse operators for embedded cell complexes [BC23]. We adjust the original implementation so that it does not assume singularities as unmovable but executes collapses and merging of singularities where prescribed by the quantization.

Block structuring Results of particularly coarse quantizations are rather suited as hex layouts than directly as hex meshes. To enable their visualization, we can refine them, yielding block-structured hex meshes with the coarse quantization serving as block structure (see, e.g., fig. 1(d)). To this end, after collapsing all zero-elements, we compute a second, now purely positive quantization (without zeros, thereby preserving the determined coarse structure), targeting a set finer resolution. We demonstrate such hex layouts and corresponding block-structured meshes in fig. 12.

Integer-grid reparametrization With the quantization values determined, an injective integer-grid map is built using per-block reparametrization as in [BC23], and then globally smoothed by minimizing the symmetric Dirichlet energy [SS15] while fixing the integer-alignment of singularities and features. Hexahedral meshes are extracted from these quantized, smoothed maps [KHB25].

6.3. Quantization Quality Measures

To assess the performance of our novel quantization method in comparison to previous methods we compare how well the quantization reaches its goal, based on two measures:

Resolution accuracy The target number of hexahedra and the actual number of hexahedra are given by the volumes V_ℓ and V_q of the input seamless map and output integer grid map, respectively. We expect the higher flexibility and higher-level objective of our method to result in higher accuracy, i.e., a generally smaller global gap between the two resolutions.

Feature distortion To also take into account local scale and alignment distortions, but only where relevant, we measure the feature distortion via (7). While other measures could be used, e.g., some notion of pointwise map distortion or hexahedral element distortion, these would inherently depend on the specific chosen subsequent map initialization and optimization post-processes. The features are exactly the parts not alterable in any kind of map-based or mesh-based post-processing, thus are most important and provide a way for generic assessment independent of further pipeline and parameter choices.

Concretely, for any two given quantizations q and q' on the same seamless input map with targets ℓ, we use the relative measures $\frac{|V_q - V_\ell| - |V_{q'} - V_\ell|}{V_\ell}$ and $\frac{d_{\text{feature}}(q',\ell)}{d_{\text{feature}}(q,\ell)}$ to assess how much more accurately q' achieves the target resolution and how much lower its feature distortion is, respectively.

The former fixed-singularity methods have a different (higher) theoretical lower limit of resolution than a flexible-singularity method. In the accuracy comparisons we wish to not punish these for their mere inability of reaching coarseness. Hence, in cases where V_ℓ is below the lowest achieved quantized volume $V_q^{\min}$ of the method on a given model (cf. table 1), this lowest-achieved volume is used as the reference point for accuracy computation instead.

6.4. Comparison

Accuracy Over all 1372 test instances, our quantization $q_{\mathbf{feat,flex}}$ is on average 17% closer to the target mesh resolution compared to the quantization $q_{\mathbf{arc,fix}}$ proposed in previous work [BBC22, BC23]. When considering only the three coarsest of our target resolution settings (10^0, 10^1, 10^2 target hexahedra), the resolution accuracy is even 30% better, as this is where flexibility of the singularity structure matters most. But even over the fine settings (10^3–10^6 target hexahedra), there still is an accuracy benefit of 7%.

When using the padding-preservation option, flexibility regarding singularity-modifying operations is obviously reduced to some extent. Nevertheless, the quantization $q_{\mathbf{feat,pad}}$ still shows an accuracy benefit over $q_{\mathbf{arc,fix}}$ of 10% (16% for the coarser and 5% for the finer target resolutions) on average over all models.

Distortion Regarding feature distortion, previous work $q_{\mathbf{arc,fix}}$ results in values higher by a average factor of 5.3 (9.0 for the coarser, 2.6 for the finer settings) compared to our new result $q_{\mathbf{feat,flex}}$. Furthermore, the feature distortion of $q_{\mathbf{arc,fix}}$ relative to our result $q_{\mathbf{feat,pad}}$ with padding-preservation is higher on average by a factor of 3.9 (6.0 for the coarser, 2.3 for the finer settings).

Timings Computing our $q_{\mathbf{feat,flex}}$ on all 1372 test instances took a total of about 18 hours single-threaded CPU time when using the GUROBI solver. With the ISP solver it took about 4 hours – costing a small optimality gap of 5%. The previous quantization routine $q_{\mathbf{arc,fix}}$ from [BBC22], on the same machine, needed a total of about 10 hours and 2 hours instead, respectively, i.e. there is a factor of around 2 in run time. Such a difference is to be expected, as our method adds flexibility, widens the search space, and employs more general structure-preservation constraints to achieve this.

6.5. Ablations

Flexible singularities To demonstrate the benefits of a flexible singular structure in isolation, we compare between fixed and flexible scenarios using the same objective function. When using our novel feature-based objective, the lack of flexible singularities in $q_{\mathbf{feat,fix}}$ results in output resolutions that are on average 12% further off the target than for $q_{\mathbf{feat,flex}}$. Its feature distortion is higher by a factor of 4.3 on average. When instead comparing scenarios using the arc-based objective, $q_{\mathbf{arc,fix}}$ is 15% further from the target resolution than $q_{\mathbf{arc,flex}}$, and arc stretching is higher by a factor of 2.1.

Another way to quantitatively assess the additional flexibility introduced by the possibility to collapse and merge singularities is to check the coarsest possible results that can be achieved (by targeting the extreme $V_\ell = 1$). Coarse quantizations are particularly interesting to be used as hex layouts, for the generation of block-structured meshes. Comparing these results, listed in table 1, quantization with flexible singularities is able to generate block layouts simpler by an order of magnitude in about a fifth of cases, and even by two orders

Table 1: *Coarsest results achievable by different methods. Shown are the minimum IGM volumes $V^{\min}$ obtained with fixed singularities, flexible singularities, or padding-preservation (and their ratios). This is equivalent to the number of blocks in the corresponding conforming block layout. Table continued in supplemental material.*

Model	$V^{\min}_{q_{\text{fix}}}$	$V^{\min}_{q_{\text{flex}}}$	$\frac{V^{\min}_{q_{\text{fix}}}}{V^{\min}_{q_{\text{flex}}}}$	$V^{\min}_{q_{\text{pad}}}$	$\frac{V^{\min}_{q_{\text{fix}}}}{V^{\min}_{q_{\text{pad}}}}$
2011 BUMPY-TORUS	612	1	612.00	7	87.43
2013 BU-HEX-OPT	305	1	305.00	7	43.57
2016 CUBESPIKES-MODEL-OUT	279	1	279.00	7	39.86
2022 ARMADILLO	774	3	258.00	17	45.53
2016 CUBESPIKES-MODEL-IN	223	1	223.00	7	31.86
2012 ROCKERARM	476	5	95.20	21	22.67
2022 ROCKERARM	441	5	88.20	21	21.00
2013 BUNNY-HEX-OPT	215	3	71.67	14	15.36
2016 ROCKERARM-HEX	621	9	69.00	15	41.40
2016 FERTILITY-HEX-LARGEL	379	6	63.17	109	3.48
2016 KITTEN-HEX	119	2	59.50	14	8.50
2022 KITTEN	119	2	59.50	14	8.50
2022 N10C-QTORUS-CYL	164	3	54.67	14	11.71
2022 N10U-QTORUS-CYL	204	4	51.00	14	14.57
2022 N09C-PYRAMID	48	1	48.00	5	9.60
2018 EXAMPLE-2	570	13	43.85	72	7.92
2014 KITTY	121	3	40.33	14	8.64
2012 ELLIPSOID-A	34	1	34.00	7	4.86
2016 HOLLOW-EIGHT-HEX	210	8	26.25	16	13.13
2011 ASM001	231	11	21.00	12	19.25
2012 DOUBLE	170	9	18.89	48	3.54
2017 EXAMPLE-3	1077	60	17.95	209	5.15
2022 N05C-BOX-MIN-PCYLS	783	48	16.31	138	5.67
2022 N05U-BOX-MIN-PCYLS	731	45	16.24	117	6.25
2022 I02U-M2	446	28	15.93	53	8.42
2022 N07C-ANTI-PYRAMID	147	10	14.70	41	3.59
2022 S06C-DODECAHEDRON	88	6	14.67	20	4.40
2019 DOUBLE-HINGE-WH	84	6	14.00	36	2.33
2022 BONE	70	5	14.00	22	3.18
2022 N08U-PENTAPYR	56	4	14.00	11	5.09
2022 S05C-CUBE-SPHERE	28	2	14.00	23	1.22
2022 S05U-CUBE-SPHERE	28	2	14.00	23	1.22
2022 S08U-CROSS-CYLS-DR	83	6	13.83	28	2.96
2022 I02C-M2	379	28	13.54	51	7.43
2022 N09U-PYRAMID	26	2	13.00	5	5.20
2022 S06B-DODECAHEDRON	811	64	12.67	454	1.79
2014 ROD	245	20	12.25	12	20.42
2022 S17C-SPHERE	12	1	12.00	7	1.71
2022 S08C-CROSS-CYLS-DR	69	6	11.50	28	2.46
2017 EXAMPLE-4	684	68	10.06	168	4.07
...					

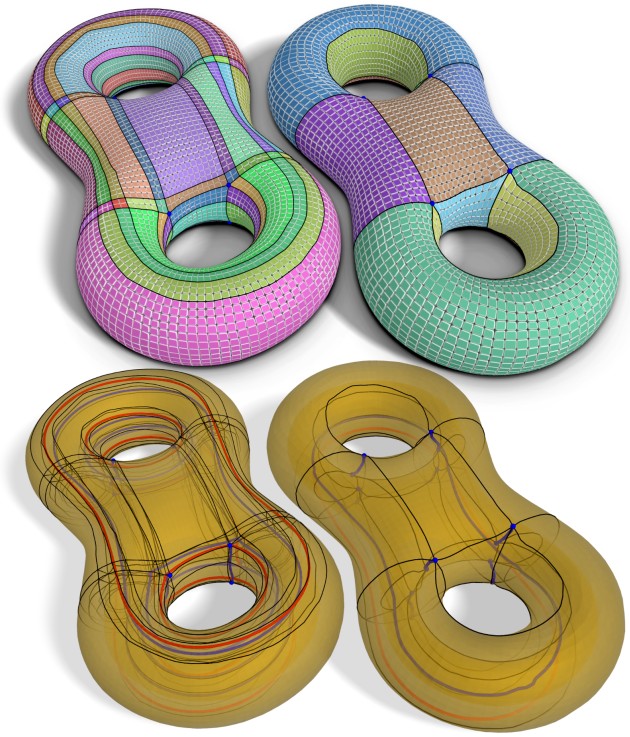


Figure 11: *Top: side-by-side comparison of the block structure of hexahedral meshes generated by the quantization $q_{\text{arc,fix}}$ from [BBC22] (left) and our quantization $q_{\text{feat,flex}}$ (right) on the same input. Bottom: side-by-side comparison of the singularity graph embedded in the block layout, with valence 3 singularities in red and valence 5 singularities in blue. $q_{\text{arc,fix}}$ keeps the original one (left), our method simplifies it in the quantization process (right).*

of magnitude in a few cases. When using the padding-preservation option, flexibility is reduced, but results are still simpler by factors of 10 or more in several cases.

Block-structured meshes generated using these coarsest block layouts are shown in fig. 12, using block layouts stemming from $q_{\text{feat,flex}}$ and from $q_{\text{arc,fix}}$, side-by-side. Block structures generated by our method are generally much simpler. Figure 11 shows another such comparison and demonstrates that the coarser block layout is indeed achieved through a significant simplification of the underlying singular structure. This simplification never comes at the expense of feature preservation, even for extreme simplifications of the singular graph, as illustrated in fig. 13.

Feature-based objective Lastly, we also investigate the impact of our feature-based objective function on quantization quality. With singularities considered fixed, $q_{\text{arc,fix}}$ misses the target resolution by 5.0% more than $q_{\text{feat,fix}}$; with flexible singularities, $q_{\text{arc,flex}}$ misses the target resolution by 2.0% more than $q_{\text{feat,flex}}$. While the overall impact of the feature-based objective function on global resolution accuracy is evidently smaller than that of the flexible singular structure analyzed above, the numbers still show some advantage of our feature-based objective in that regard.

Its main virtue, however, is that it more faithfully preserves the spacing and sizing of features compared to the unnecessarily fine-grained arc-based objective. Indeed, for both fixed and flexible singularities, using the old arc-based objective yields a feature distortion that is higher by about 60% than when using our novel feature-based objective, showing that the distortion can often be improved if individual arcs can deviate further from their individual targets. Figure 14 shows examples, where the feature-based objective is able to not only match the target resolution much more closely, but also maintains a globally consistent spacing and sizing of features.

7. Conclusion

We have introduced a quantization routine for volumetric seamless parametrizations that adaptively simplifies the underlying singular structure, depending on the target mesh resolution, while accurately preserving shape and features in the output integer-grid map. Applying this quantization in a hexahedral meshing pipeline, where former approaches fixed the singular structure in an early step and prevented readjustments, we showed that unfreezing the

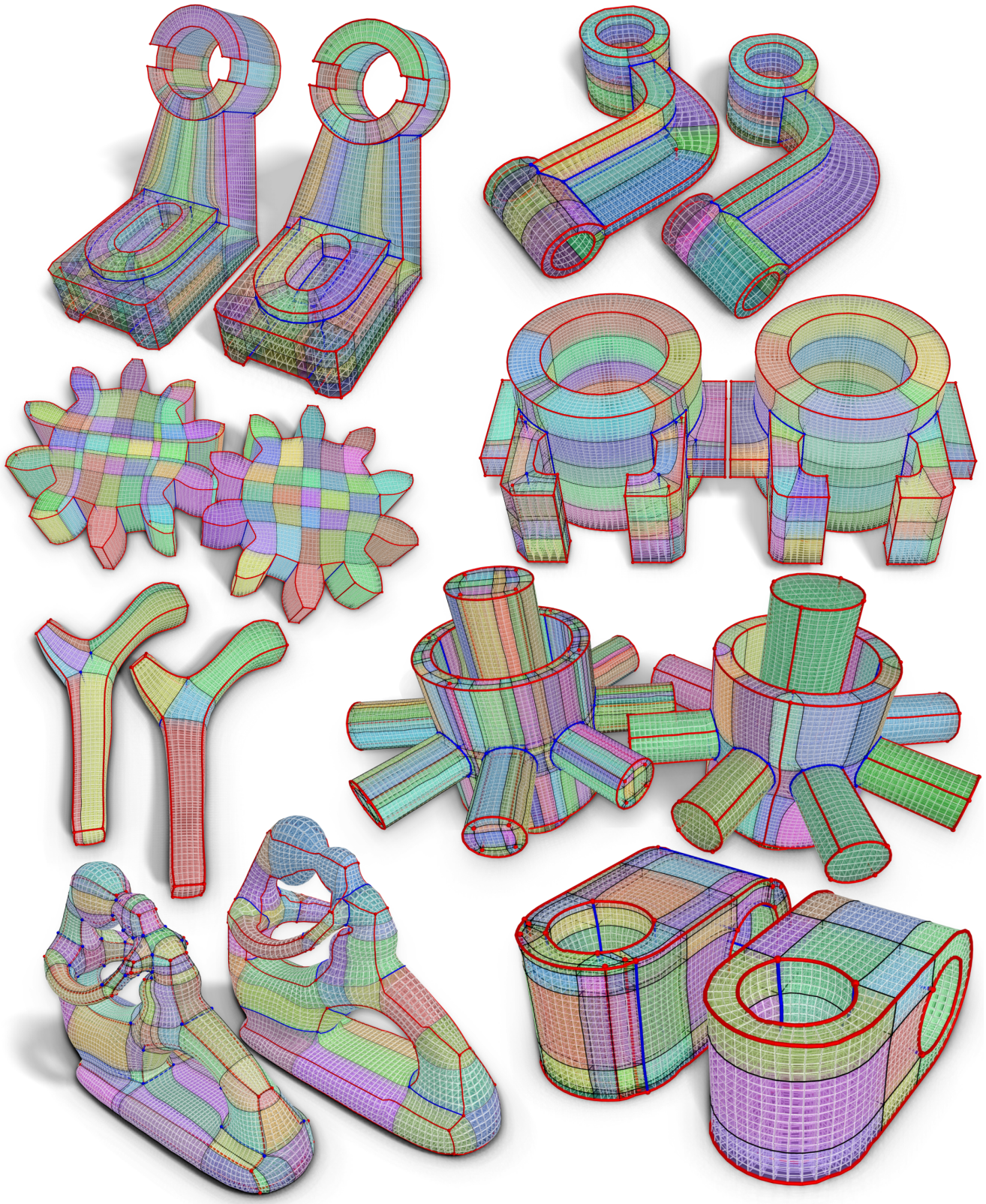


Figure 12: *Side-by-side comparison of the block structure of hexahedral meshes generated by the quantization $q_{arc,fix}$ from [BBC22] (left) and our quantization $q_{feat,flex}$ (right) on the same inputs. Valence 3 singularities are shown in red, valence 5 ones in blue.*

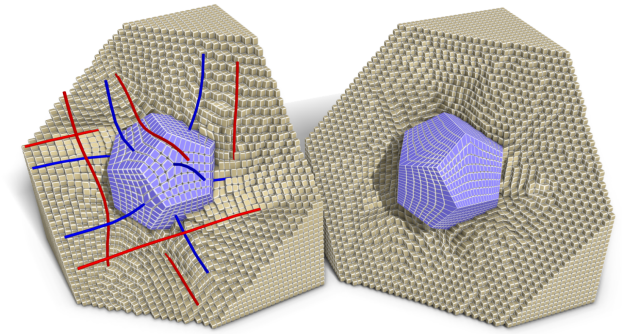


Figure 13: *Cut-away view of hexahedral meshes generated for a cube model with internal feature surface (purple) by our method, targeting a finer block layout (left) or the coarsest possible block layout (right). Valence 3 (red) and 5 (blue) singularities, vanish through merging in the coarser quantization. In contrast to simplification methods operating as a post-process on the final hexahedral mesh, structure simplifications and the exact preservation of domain and features are completely decoupled, so both can be achieved.*

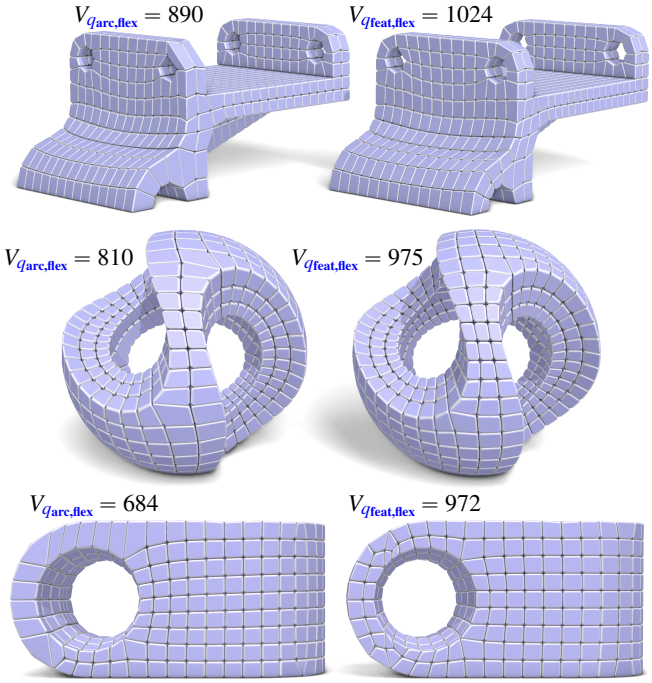


Figure 14: *Some models, each with a target volume of $V_\ell = 1000$, quantized and meshed with flexible singularities and (left) the arc-based objective or (right) the feature-based objective. Our feature-based objective function is able to maintain a more uniform resolution and globally consistent spacing and sizing of features.*

singular structure leads to improved resolution uniformity and an often preferable simpler block structure of the generated meshes. We furthermore demonstrated in an ablation study that both core ingredients of our approach – flexible structure-preserving constraints and a feature-based objective function – are relevant for achieving these improvements.

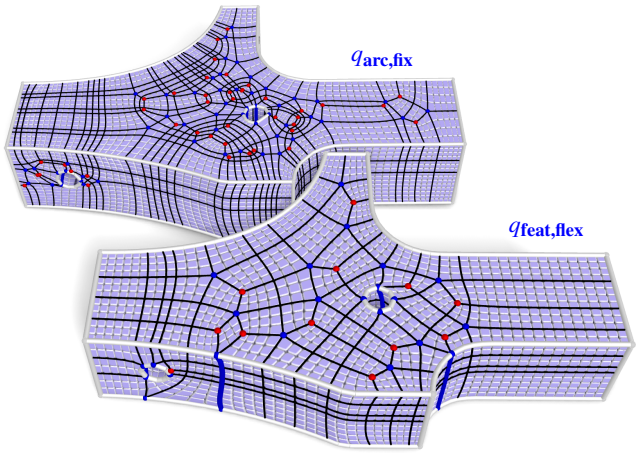


Figure 15: *On the same input instance, which exhibits an overly complex singularity structure (86 non-feature singularities), our new method (bottom) achieves a substantial simplification (down to 27 non-feature singularities) and hence a much simpler block layout than previous methods that adhere to the original singularities (top). Note, however, that despite a very coarse layout being targeted the result still contains significantly more singularities than ideally desired, due to singularity cluster index constraints being greedy.*

7.1. Limitations and Future Work Directions

Singularity positions While our method is able to simplify the singular structure by removing and recombining singularities, it does not adjust the object space positions of the singularities that remain. Especially when significant simplifications were performed, these positions are often suboptimal for overall map distortion. While they can be optimized as part of a subsequent hex mesh optimizaton, a more principled approach would be addressing this already on the parametrization level, via an optimization that shifts singularities to minimize the underlying parametrization distortion, as has been done for surface singularities [CK14]. The added complexity of the volumetric setting, especially regarding the interplay of continuous positions and discrete mesh embedding and the general difficulty of robust volumetric parametrization, make this an interesting task for future work.

Solution cut-off The linear structure-preservation constraints of the core integer quadratic program are conservative: they cut off not only the invalid solutions but also some valid ones. This slight over-restriction of the solution space is common to recent quantization methods, but is more obvious for our constraints that prevent singularity clusters with problematic indices; our constraints will always force a problematic cluster to split rather than to possibly recombine with another cluster. While this choice is safe in terms of validity, it sometimes leads to suboptimal results in areas where the input singularity structure is particularly messy. Such an example is shown in fig. 15, with singular structure before and after our quantization-driven simplification. While the simplification is quite strong, the final singular structure still is distant from maximal simplicity. Exploring different branches induced by alternative linear constraints or opting for more general non-linear constraints could improve this.

Balancing goals While here the main objective was reaching a desired mesh resolution, while keeping feature distortion as low as possible under this condition, a different balancing of these often opposing aspects of simplicity and low distortion can also be highly interesting. Employing a two-step quantization process to that end, like we did for the block-structuring experiments (the first to determine a simple block layout, the second for matching the desired mesh resolution), is also commonly employed in quad meshing [LCK21a, LCK21b], but involves educated guessing of parameters that determine the balance of layout coarseness and quality only indirectly. From the perspective of an end-user it would be preferable to have more intuitive and predictable control over this.

Acknowledgments

Open Access funding enabled and organized by Projekt DEAL.

References

[BBC22] Brückler H., Bommes D., Campen M.: Volume parametrization quantization for hexahedral meshing. *ACM Transactions on Graphics 41*, 4 (2022). 3, 4, 5, 6, 7, 8, 12, 13, 14, 15

[BBC24] Brückler H., Bommes D., Campen M.: Integer-sheet-pump quantization for hexahedral meshing. *Computer Graphics Forum 43*, 5 (2024), e15131. 2, 3, 4, 5, 12

[BBS02] Borden M. J., Benzley S. E., Shepherd J. F.: Hexahedral sheet extraction. In *Proceedings of the 11th International Meshing Roundtable* (2002), pp. 147–152. 4

[BC23] Brückler H., Campen M.: Collapsing embedded cell complexes for safer hexahedral meshing. *ACM Transactions on Graphics 42*, 6 (2023). 2, 3, 4, 12, 13

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

[BGMC22] Brückler H., Gupta O., Mandad M., Campen M.: The 3d motorcycle complex for structured volume decomposition. *Computer Graphics Forum 41*, 2 (2022), 221–235. 2, 4, 12

[Bla01] Blacker T.: Automated conformal hexahedral meshing constraints, challenges and opportunities. *Engineering with Computers 17*, 3 (2001), 201–210. 1

[BPM*95] Benzley S. E., Perry E., Merkley K., Clark B., Sjaardema G.: A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings of the 4th International Meshing Roundtable* (1995), pp. 179–191. 1

[BTPB07] Bourdin X., Trosseille X., Petit P., Beillas P.: Comparison of tetrahedral and hexahedral meshes for organ finite element modeling: an application to kidney impact. In *Proceedings of the 20th Int. Technical Conference on the Enhanced Safety of Vehicles* (2007). 1

[BZK09] Bommes D., Zimmer H., Kobbelt L.: Mixed-integer quadrangulation. *ACM Transactions on Graphics 28*, 3 (2009). 3

[CAS*19] Cherchi G., Alliez P., Scateni R., Lyon M., Bommes D.: Selective padding for polycube-based hexahedral meshing. *Computer Graphics Forum 38*, 1 (2019), 580–591. 1

[CBK15] Campen M., Bommes D., Kobbelt L.: Quantized global parametrization. *ACM Transactions on Graphics 34*, 6 (2015). 3, 5

[CK92] Cifuentes A., Kalbag A.: A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. *Finite Elements in Analysis and Design 12*, 3 (1992), 313 – 318. 1

[CK14] Campen M., Kobbelt L.: Quad layout embedding via aligned parameterization. *Computer Graphics Forum 33*, 8 (2014), 69–81. 16

[CODH*24] Coudert-Osmont Y., Desobry D., Heistermann M., Bommes D., Ray N., Sokolov D.: Quad mesh quantization without a t-mesh. *Computer Graphics Forum 43*, 1 (2024), e14928. 4, 5, 10

[EGKT08] Eppstein D., Goodrich M. T., Kim E., Tamstorf R.: Motorcycle graphs: canonical quad mesh partitioning. *Comp. Graph. Forum 27*, 5 (2008), 1477–1486. 3

[Epp03] Eppstein D.: Dynamic generators of topologically embedded graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 2003), SODA '03, Society for Industrial and Applied Mathematics, pp. 599–608. 9

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

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

[GPW*17] Gao X., Panozzo D., Wang W., Deng Z., Chen G.: Robust structure simplification for hex re-meshing. *ACM Transactions on Graphics 36*, 6 (2017). 4

[JHW*14] Jiang T., Huang J., Wang Y., Tong Y., Bao H.: Frame field singularity correction for automatic hexahedralization. *IEEE Transactions on Visualization and Computer Graphics 20*, 8 (2014), 1189–1199. 4

[KHB25] Kohler T., Heistermann M., Bommes D.: HexHex: High-speed extraction of hexahedral meshes. *ACM Transactions on Graphics 44*, 4 (2025). 2, 13

[KLSO12] Kowalski N., Ledoux F., Staten M. L., Owen S. J.: Fun sheet matching: towards automatic block decomposition for hexahedral meshes. *Engineering with Computers 28*, 3 (2012), 241–253. 4

[KNP07] Kälberer F., Nieser M., Polthier K.: QuadCover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (2007), 375–384. 3

[LB23] Liu H., Bommes D.: Locally meshable frame fields. *ACM Transactions on Graphics 42*, 4 (2023). 2, 4

[LCBK19] Lyon M., Campen M., Bommes D., Kobbelt L.: Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Transactions on Graphics 38*, 4 (2019). 3

[LCK21a] Lyon M., Campen M., Kobbelt L.: Quad layouts via constrained t-mesh quantization. *Computer Graphics Forum 40*, 2 (2021), 305–314. 4, 17

[LCK21b] Lyon M., Campen M., Kobbelt L.: Simpler quad layouts using relaxed singularities. *Computer Graphics Forum 40*, 5 (2021), 169–179. 2, 4, 17

[LS10] Ledoux F., Shepherd J.: Topological modifications of hexahedral meshes via sheet operations: a theoretical study. *Engineering with Computers 26*, 4 (2010), 433–447. 4

[LZC*18] Liu H., Zhang P., Chien E., Solomon J., Bommes D.: Singularity-constrained octahedral fields for hexahedral meshing. *ACM Transactions on Graphics 37*, 4 (2018). 4

[Mit00] Mitchell S. A.: High fidelity interval assignment. *International Journal of Computational Geometry & Applications 10*, 4 (2000), 399–415. 3

[Mit14] Mitchell S. A.: Simple and fast interval assignment using nonlinear and piecewise linear objectives. In *Proceedings of the 22nd International Meshing Roundtable*, Sarrate J., Staten M., (Eds.). Springer, 2014, pp. 203–221. 3

[Mit23] Mitchell S. A.: Incremental interval assignment by integer linear algebra with improvements. *Computer-Aided Design 158* (2023), 103485. 3

[MPZ14] Myles A., Pietroni N., Zorin D.: Robust field-aligned global parametrization. *ACM Transactions on Graphics 33*, 4 (2014). 4

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: Cubecover – parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406. 3, 4

[PCS*22] PIETRONI N., CAMPEN M., SHEFFER A., CHERCHI G., BOMMES D., GAO X., SCATENI R., LEDOUX F., REMACLE J., LIVESU M.: Hex-mesh generation and processing: A survey. *ACM Transactions on Graphics 42*, 2 (2022). 2, 3

[PNA*21] PIETRONI N., NUVOLI S., ALDERIGHI T., CIGNONI P., TARINI M.: Reliable feature-line driven quad-remeshing. *ACM Transactions on Graphics 40*, 4 (2021). 4

[PPM*16] PIETRONI N., PUPPO E., MARCIAS G., SCOPIGNO R., CIGNONI P.: Tracing field-coherent quad layouts. *Computer Graphics Forum 35*, 7 (2016), 485–496. 4

[RCR19] REBEROL M., CHEMIN A., REMACLE J.-F.: Multiple approaches to frame field correction for cad models. In *Proceedings of the 28th International Meshing Roundtable* (2019), pp. 283–295. 4

[RS06] RAMOS A., SIMÕES J.: Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur. *Medical Engineering & Physics 28*, 9 (2006), 916 – 924. 1

[SRRGRN14] SARRATE RAMOS J., RUIZ-GIRONÉS E., ROCA NAVARRO F. J.: Unstructured and semi-structured hexahedral mesh generation methods. *Computational Technology Reviews 10* (2014), 35–64. 1

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Transactions on Graphics 34*, 4 (2015). 13

[SZC*22] SHEN H., ZHU L., CAPOUELLEZ R., PANOZZO D., CAMPEN M., ZORIN D.: Which cross fields can be quadrangulated? global parameterization from prescribed holonomy signatures. *ACM Transactions on Graphics 41*, 4 (2022), 1–12. 4

[TA93] TAM T., ARMSTRONG C. G.: Finite element mesh control by integer programming. *International Journal for Numerical Methods in Engineering 36*, 15 (1993), 2581–2605. 3

[TEC11] TADEPALLI S. C., ERDEMIR A., CAVANAGH P. R.: Comparison of hexahedral and tetrahedral elements in finite element analysis of the foot and footwear. *Journal of Biomechanics 44*, 12 (2011), 2337 – 2343. 1

[TK03] TAUTGES T. J., KNOOP S. E.: Topology modification of hexahedral meshes using atomic dual-based operations. In *Proceedings of the 12th International Meshing Roundtable* (2003), pp. 415–423. 4

[VCV25] VEKHTER J., CHEN Z., VOUGA E.: Mint: Discretely integrable moments for symmetric frame fields. *Computer Graphics Forum 44*, 5 (2025), e70193. 2

[WCO21] WANG W., CAO Y., OKAZE T.: Comparison of hexahedral, tetrahedral and polyhedral cells for reproducing the wind field around an isolated building by LES. *Building and Environment 195* (2021), 107717. 1

[WNR04] WANG E., NELSON T., RAUCH R.: Back to elements-tetrahedra vs. hexahedra. In *Proceedings of the 2004 international ANSYS conference* (2004). 1

[XLZ*21] XU G., LING R., ZHANG Y. J., XIAO Z., JI Z., RABCZUK T.: Singularity structure simplification of hexahedral meshes via weighted ranking. *Computer-Aided Design 130* (2021), 102946. 4

[ZCFM23] ZHANG P., CHIANG J. H.-H., FAN X. C., MUNDILOVA K.: Local decomposition of hexahedral singular nodes into singular curves. *Computer-Aided Design 158* (2023), 103484. 4

[ZDLL25] ZHENG X., DUAN J., LEI N., LUO Z.: Feature-aware singularity structure optimization for hex mesh. *Computer-Aided Design 180* (2025), 103825. 4

Appendix A: Modified Integer-Sheet-Pump

The original integer-sheet pump algorithm is tailored specifically to the arc-based objective from eq. (3). First a greedy heuristic orders the arc variables q_i by how negatively they impact $d(q,\ell)$, then for each q_i in this order a linear program tries to find a minimal, constraint-feasible step that would update q_i and decrease $d(q,\ell)$ (or at least increase it as little as possible). If this step is required (to establish constraint satisfaction) or decreases d, it is executed and the process repeated until no more such step can be found.

The LP that determines feasible steps (named *integer-sheets*) has two variables from $\mathbb{R}_0^+$ for each q_i, modeling its increase $\Delta^+ q_i$ and decrease $\Delta^- q_i$. The constraints of the LP are identical to the constraints in eq. (2), i.e. the LP only admits steps that establish or maintain feasibility. In the linear objective minimized by the LP each variable is weighted by a purely positive *unfavorability* weight: a high unfavorability is assigned when an isolated $\Delta^\pm q_i = 1$ would increase d, a medium unfavorability if d would be almost unaffected and a low unfavorability if d would decrease. The objective-optimal step determined by the LP then is one that affects as few variables as possible and preferably ones that increase d as little as possible.

In the original work, the way the ordering and weighting is chosen exploited the fact that $\mathcal{F}$ in the global objective function (4) was the identity. In that case the energy landscape is isotropic, and arc variables are independent from one another (barring constraints). In that setting, moving each arc closer to its global target is optimal not only globally but also locally. In other words: the vector pointing towards the global optimum and the negative gradient align. This effectively simplified three aspects of the algorithm:

1. The arc ordering can be done along $|q_i - \ell_i|$,
2. LP weights can be tiered according to $\pm(q_i - \ell_i)$,
3. a chain of single-coordinate descent steps (barring constraints) leads to the global optimum.

In our case, where $\mathcal{F}$ is not the identity matrix, we have considered several more general strategies for each of these aspects:

1. The arc ordering could remain based on $|q_i - \ell_i|$, or be based on the gradient $\nabla d(q,\ell)$ (concretely $\frac{\partial d}{\partial q_i}$) or be based on $|q_i - q_i^*|$ where q_i^* is the optimum of q_i if all other variables are assumed fixed at their current values.
2. The three tier LP weighting system could remain based on $\pm(q_i - \ell_i)$, or be based on $\pm(q_i - q_i^*)$, or a combination thereof, e.g. with hierarchical tiers or tiers based on min/max/average of the two.
3. The greedy descent can get stuck in local minima where two arc variables q_i, q_j that are dependent via $\mathcal{F}$ but independent via constraints would have to be simultaneously incremented and decremented, respectively. To escape from such local minima, additional computation time could be invested to determine minimal two-variable updates.

In our experiments we have determined that a favorable balance between speed and optimality is achieved by keeping the ordering of arcs as is, employing the normal LP weighting by default and only when that LP does not find a descent step, repeat the LP computation with a weighting based in the same way on $\max(\pm(q_i - \ell_i), \pm(q_i - q_i^*))$. Furthermore, once no more descent step can be found, we determine minimal two-variable updates by the same LP formulation for each pair of arcs dependent via $\mathcal{F}$. Note that these dependencies are quite sparse for our $\mathcal{F}$. These modifications of the default ISP method improve the average optimality gap on our instances from about 20% to 5%, while only increasing run times by around 2×.

Table 2: *Coarsest possible integer-grid map volume with fixed, flexible or boundary-padded singularities (continued from table 1)*

Model	$V^{\min}_{q_{\text{fix}}}$	$V^{\min}_{q_{\text{flex}}}$	$\frac{V^{\min}_{q_{\text{fix}}}}{V^{\min}_{q_{\text{flex}}}}$	$V^{\min}_{q_{\text{pad}}}$	$\frac{V^{\min}_{q_{\text{fix}}}}{V^{\min}_{q_{\text{pad}}}}$
…					
2022 N03C-SKIJUMP-BOX-CYL	30	3	10	24	1.25
2022 N03U-SKIJUMP-BOX-CYL	30	3	10	24	1.25
2022 N06U-ANTI-PENTAPYR	107	11	9.73	48	2.23
2022 I10U-SIMP	126	13	9.69	29	4.34
2017 EXAMPLE-2	463	48	9.65	127	3.65
2016 ROTELLIPSE-PADDED	19	2	9.5	7	2.71
2016 TWISTEDU	19	2	9.5	7	2.71
2022 N08C-PENTAPYR	56	6	9.33	11	5.09
2022 N04U-TRANSITION-PRISM	27	3	9	12	2.25
2022 N06C-ANTI-PENTAPYR	99	11	9	33	3
2022 N12C-LIMIT-CYCLE-GENUS0	9	1	9	6	1.5
2022 S11C-CUBE-CYL	27	3	9	22	1.23
2022 S11U-CUBE-CYL	27	3	9	22	1.23
2022 S12C-CUBE-ROUNDED-1	81	9	9	39	2.08
2022 S12U-CUBE-ROUNDED-1	81	9	9	39	2.08
2022 S13C-CUBE-ROUNDED-2	81	9	9	39	2.08
2022 S13U-CUBE-ROUNDED-2	81	9	9	39	2.08
2022 S17B-SPHERE	56	7	8	56	1
2022 CAMILLE-HAND	85	11	7.73	48	1.77
2022 I13C-S6	855	111	7.7	349	2.45
2022 I13U-S6	725	95	7.63	347	2.09
2022 I14C-S7	64	9	7.11	20	3.2
2022 I14U-S7	64	9	7.11	20	3.2
2012 ELLIPSOID-B	7	1	7	7	1
2012 ELLIPSOID-C	7	1	7	7	1
2022 S06U-DODECAHEDRON	42	6	7	20	2.1
2022 SPHERE	7	1	7	7	1
2019 DOUBLE-HINGE-NH	165	27	6.11	45	3.67
2016 KNOT-HEX	60	10	6	50	1.2
2022 S04B-TETRAHEDRON	120	20	6	53	2.26
2022 S13B-CUBE-ROUNDED-2	233	39	5.97	135	1.73
2022 I09U-M9	652	111	5.87	192	3.4
2022 I09C-M9	554	104	5.33	192	2.89
2022 I10C-SIMP	119	23	5.17	55	2.16
2022 S08B-CROSS-CYLS-DR	211	41	5.15	1034	0.2
2022 I18C-S22	192	38	5.05	73	2.63
2016 KPDLOEKR-HEX	230	46	5	230	1
2022 CUBE-SPHERE	10	2	5	10	1
2022 CYLINDER	5	1	5	5	1
2022 S14C-CUBE-CORNER-SUB-SPHERE	10	2	5	10	1
2022 S14U-CUBE-CORNER-SUB-SPHERE	10	2	5	10	1
2022 S15C-CYLINDER	10	2	5	10	1
2022 S15U-CYLINDER	10	2	5	10	1
2022 S16C-TORUS	35	7	5	35	1
2022 I23U-S31	464	95	4.88	145	3.2
2019 BROKENBULLET	24	5	4.8	17	1.41
2022 BROKEN-BULLET	24	5	4.8	17	1.41
2022 I25U-S40	314	66	4.76	159	1.97
2022 I23C-S31	428	93	4.6	149	2.87
2012 ROD	50	11	4.55	35	1.43
2022 N12B-LIMIT-CYCLE-GENUS0	108	24	4.5	141	0.77
2022 N07U-ANTI-PYRAMID	66	15	4.4	37	1.78
2022 I22C-S27	326	76	4.29	147	2.22
2022 I25C-S40	337	80	4.21	164	2.05
2022 I18U-S22	152	38	4	88	1.73
2022 N04C-TRANSITION-PRISM	8	2	4	8	1
2022 TETRAHEDRON	4	1	4	4	1
2019 FANDISK	53	14	3.79	40	1.33
2022 I12U-S5	246	65	3.78	130	1.89
2022 I12C-S5	268	72	3.72	166	1.61
2022 N04B-TRANSITION-PRISM	78	21	3.71	96	0.81
2019 FANDISK.LIU18	51	14	3.64	40	1.28
2017 JOINT	54	15	3.6	38	1.42
2022 S05B-CUBE-SPHERE	126	36	3.5	160	0.79
2022 I01U-M1	213	61	3.49	84	2.54
2022 S10C-CYL-CUTSPHERE	24	7	3.43	17	1.41
2022 S10U-CYL-CUTSPHERE	24	7	3.43	17	1.41
2022 S15B-CYLINDER	51	15	3.4	132	0.39
2012 JOINT	54	17	3.18	32	1.69
2017 PONE.0177603.S003	54	17	3.18	32	1.69
2012 SCULPTURE-B	28	9	3.11	14	2
2022 FANDISK	42	14	3	34	1.24
2022 N12U-LIMIT-CYCLE-GENUS0	9	3	3	6	1.5
2022 S02C-PRISM	3	1	3	3	1
2022 S02U-PRISM	3	1	3	3	1
2022 I11C-S1	49	17	2.88	26	1.88
…					

Table 3: *Continued from table 2*

Model	$V^{\min}_{q_{\text{fix}}}$	$V^{\min}_{q_{\text{flex}}}$	$\frac{V^{\min}_{q_{\text{fix}}}}{V^{\min}_{q_{\text{flex}}}}$	$V^{\min}_{q_{\text{pad}}}$	$\frac{V^{\min}_{q_{\text{fix}}}}{V^{\min}_{q_{\text{pad}}}}$
…					
2022 JOINT	49	17	2.88	32	1.53
2022 N02U-SKIJUMP-ANTI-BOX-CYL	14	5	2.8	8	1.75
2022 I20U-S25	166	60	2.77	74	2.24
2022 S02B-PRISM	66	24	2.75	39	1.69
2022 I11U-S1	46	17	2.71	27	1.7
2019 CHAMFER-L4	8	3	2.67	8	1
2022 S09C-BRIDGE	8	3	2.67	6	1.33
2022 S09U-BRIDGE	8	3	2.67	6	1.33
2022 I20C-S25	163	62	2.63	84	1.94
2022 S07C-NOTCH	13	5	2.6	11	1.18
2022 S07U-NOTCH	13	5	2.6	11	1.18
2022 S11B-CUBE-CYL	132	52	2.54	424	0.31
2022 S14B-CUBE-CORNER-SUB-SPHERE	88	35	2.51	88	1
2022 N13C-ACUTE-LINE	10	4	2.5	10	1
2022 N13U-ACUTE-LINE	10	4	2.5	10	1
2022 S03C-PENTAGON	5	2	2.5	5	1
2022 S03U-PENTAGON	5	2	2.5	5	1
2019 GEAR	62	25	2.48	45	1.38
2022 I22U-S27	184	76	2.42	132	1.39
2017 FEMUR-SHELL0	12	5	2.4	10	1.2
2022 N02C-SKIJUMP-ANTI-BOX-CYL	12	5	2.4	8	1.5
2022 S03B-PENTAGON	84	36	2.33	57	1.47
2022 I15C-S8	151	69	2.19	116	1.3
2022 I15U-S8	151	69	2.19	116	1.3
2019 WRENCH	45	21	2.14	27	1.67
2022 S09B-BRIDGE	90	42	2.14	90	1
2022 N13B-ACUTE-LINE	51	24	2.13	51	1
2022 I06U-M6	105	51	2.06	80	1.31
2022 I06C-M6	101	50	2.02	80	1.26
2016 ROCKERARM-POLYCUBE-OUT	211	109	1.94	109	1.94
2022 S07B-NOTCH	96	52	1.85	120	0.8
2015 IMPELLER-STRESSTEST-OUT	272	148	1.84	200	1.36
2019 COLUMN	18	10	1.8	18	1
2012 SCULPTURE-A	16	9	1.78	10	1.6
2022 SCULPTURE	16	9	1.78	8	2
2017 FANDISK	39	22	1.77	28	1.39
2017 PONE.0177603.S002	39	22	1.77	28	1.39
2022 FANPART	5	3	1.67	5	1
2016 TEAPOT-POLYCUBE-OUT	92	56	1.64	56	1.64
2016 BLOCK-POLYCUBE-IN	35	24	1.46	24	1.46
2016 FEMUR-POLYCUBE-IN	37	27	1.37	27	1.37
2016 FEMUR-POLYCUBE-OUT	37	27	1.37	27	1.37
2012 FANDISK	30	22	1.36	28	1.07
2016 TABLE1-POLYCUBE-OUT	53	39	1.36	39	1.36
2016 TABLE1-POLYCUBE-IN	57	43	1.33	43	1.33
2016 BUNNY-POLYCUBE-IN	46	36	1.28	36	1.28
2016 JOINT-HEX	47	39	1.21	39	1.21
2016 CHINESE-DRAGON-POLYCUBE-OUT	85	73	1.16	73	1.16
2017 FEMUR-SHELL1	22	19	1.16	19	1.16
2012 HANGER	46	40	1.15	40	1.15
2015 HANGER-STRESSTEST-OUT	46	40	1.15	40	1.15
2019 HANGER	46	40	1.15	40	1.15
2016 BUNNY-POLYCUBE-OUT	40	36	1.11	36	1.11
2016 ROCKERARM-POLYCUBE-IN	115	107	1.07	107	1.07
2016 CHINESE-DRAGON-POLYCUBE-IN	75	74	1.01	74	1.01
2016 ASM-POLYCUBE-IN	41	41	1	41	1
2016 ASM-POLYCUBE-OUT	41	41	1	41	1
2016 BLOCK-POLYCUBE-OUT	24	24	1	24	1
2016 CUBESPIKES-POLYCUBE-IN	33	33	1	33	1
2016 CUBESPIKES-POLYCUBE-OUT	33	33	1	33	1
2016 FANCY-RING-HEX	30	30	1	30	1
2016 FANDISK-POLYCUBE-IN	52	52	1	52	1
2016 FANDISK-POLYCUBE-OUT	52	52	1	52	1
2016 HAND-POLYCUBE-OUT	25	25	1	25	1
2016 NUT-HEX	12	12	1	12	1
2016 TABLE2-POLYCUBE-IN	20	20	1	20	1
2016 TABLE2-POLYCUBE-OUT	20	20	1	20	1
2016 TEAPOT-POLYCUBE-IN	56	56	1	56	1
2017 CUBE	17	17	1	17	1
2017 CYLINDER-GRID	2	2	1	2	1
2017 PONE.0177603.S001	17	17	1	17	1
2018 EXAMPLE-5	18	18	1	18	1
2019 DOUBLE-TORUS	11	11	1	11	1
2019 TETRAHEDRON	4	4	1	4	1
2022 S01B-CUBE	27	27	1	27	1
2022 S01C-CUBE	1	1	1	1	1
2022 S01U-CUBE	1	1	1	1	1
2022 S04C-TETRAHEDRON	4	4	1	4	1
2022 S04U-TETRAHEDRON	4	4	1	4	1
2016 HAND-POLYCUBE-IN	23	23	1.00	23	1.00