Intrinsic Mixed-Integer Polycubes for Hexahedral Meshing

Manish Mandad[a], Ruizhi Chen[b], David Bommes[c], Marcel Campen[a]

[a] *Osnabrück University, Germany*
[b] *RWTH Aachen University, Germany*
[c] *University of Bern, Switzerland*

Abstract

Polycube mapping is an attractive approach for the generation of all-hexahedral meshes with a fully regular interior, i.e. free of internal singular edges or vertices. It is based on determining a low distortion map between the input model and a polycube domain, which then pulls back the regular voxel grid to form a hexahedral mesh for the model. Automatically finding an appropriate polycube domain for a given model, however, is a challenging problem. Existing algorithms are either very sensitive to the embedding and orientation of the input model, restricted to only subclasses of possible domains, or depend crucially on some initialization because they rely on a non-convex optimization formulation. This can easily lead to unsatisfactory and unnecessary corners and edges in the polycube structure. We present a novel approach to the problem of finding high-quality polycube domains. It is based on an entirely intrinsic formulation as a mixed integer optimization problem, which can be tackled by solving a series of simple convex problems, each of which can be solved to the global optimum. Experiments demonstrate that our method avoids many of the undesired corners and surface irregularities common to many previous methods.

Keywords: Polycube mapping, Hexahedral meshing, Volumetric deformation

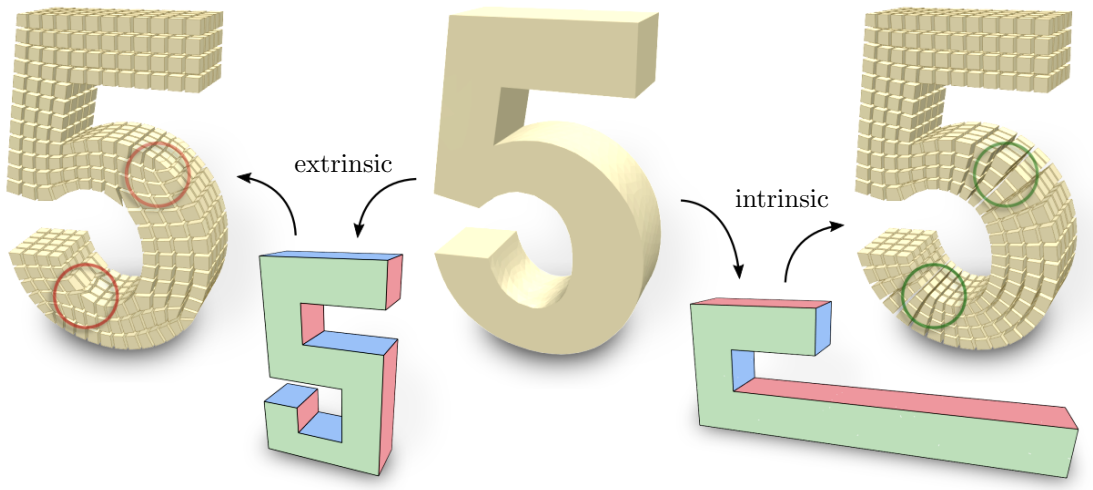


Figure 1: Hexahedral meshes generated for a given model (center) via maps to polycube domains (bottom) which have been generated either extrinsically (left) or intrinsically (right). Notice how the extrinsic approach creates numerous spurious, unnecessary corners and kinks in the polycube domain, which translate into extraordinary boundary vertices and edges in the mesh (some circled), often lowering mesh quality unnecessarily.

Email addresses: manishmandad@gmail.com (Manish Mandad), david.bommes@inf.unibe.ch (David Bommes), campen@uos.de (Marcel Campen)

Preprint submitted to GMP 2022

1. Introduction

Manually or semi-manually designed hexahedral volume meshes have been used for decades in computer-aided engineering, simulation, and analysis. The significantly higher cost compared to tetrahedral mesh generation, which can be robustly automated, may be justified by higher efficiency and accuracy observed in various fields (Sarrate et al., 2014; Cifuentes and Kalbag, 1992; Benzley et al., 1995; Ramos and Simões, 2006; Bourdin et al., 2007; Tadepalli et al., 2011), though not in general (Schneider et al., 2019).

Following advances in the automatic generation of quadrilateral surface meshes (Kälberer et al., 2007; Bommes et al., 2009, 2013a) based on field-guided integer grid maps, recent research has looked into extending these results for the automatic generation of high-quality volumetric hexahedral meshes. Unfortunately, the generalization of the involved 2D cross fields to 3D frame fields comes with a number of crucial theoretical and practical problems (Vaxman et al., 2016). Unless one resorts to manual efforts once again (Nieser et al., 2011), the results of field generation (Huang et al., 2011) are thus not rarely invalid. Problems are related to the singularities of the fields, which in the 3D case do not generally translate into a proper structure of irregular (or *extraordinary*) edges and vertices for the hexahedral mesh (Liu et al., 2018).

Efforts have been spent to eliminate invalid configurations in a post-process (Huang et al., 2012; Li et al., 2012; Jiang et al., 2014; Reberol et al., 2019). While these methods are able to fix local issues in the field, global inconsistencies remain. Hence, the problem of automatically generating a 3D frame field with a free, non-prescribed singularity structure that is guaranteed to imply a valid irregularity structure for a hexahedral mesh is still open, preventing this class of algorithms from being practically sufficient yet.

Interior-Regular Hexahedral Meshes

An attractive alternative, which avoids the aforementioned problem by design, is to restrict to the class of *interior-regular* hexahedral meshes, i.e. those with irregular edges and vertices only on the boundary but not in the interior. Such meshes can be obtained by constructing a polycube map, mapping the input model to an axis-aligned polycube domain, such that the inverse map turns part of the voxel grid into a boundary aligned hexahedral mesh for the model (where the cubes of the polycube correspond to the individual hexahedra of the mesh), see Figure 1. This approach is attractive because, with proper definition of the employed map (cf. Section 3), it is fully general, i.e. theoretically able to generate *any* interior-regular hexahedral mesh for a given solid model.

The restriction to interior-regular meshes can put some constraints on the mesh quality (in terms of element shape) that can be achieved. At the same time, it can also be an advantage in certain contexts where internal irregularities have significant adverse effects, e.g., due to reduced smoothness in volumetric spline space constructions (Wang et al., 2013; Wei et al., 2018).

The key challenge in this context is finding a suitable polycube domain and finding a bijective map to that domain.

Previous approaches to this challenge can be classified into two categories: separate generation of the domain and the map into this domain in two steps, or combined generation of both in an integrated manner. In the latter case the map quality, thus the resulting mesh quality, can directly influence the domain structure, whereas in the former case such geometric information is not or only heuristically available in the domain construction process.

Contribution

Our method belongs to the more powerful class of combined construction of map and domain. We point out that this combined problem is effectively a mixed integer optimization problem: there are continuous and discrete degrees of freedom, the discrete ones stemming from the fact that the domain boundary needs to be piecewise aligned to a discrete set of (axis-aligned) orientations in order to form a polycube, i.e. in order to imply a hexahedral mesh whose elements are properly aligned with the model's boundary.

Such problems are notoriously hard to solve to the global optimum. Recently, efficient approximative strategies have been used successfully for mesh generation purposes: via *successive* discretization (or *rounding*) of a continuous relaxation of the mixed integer problem (Bommes et al., 2010), high quality results can be obtained in the construction of direction fields, quad meshes, and hex meshes (Bommes et al., 2009;

Li et al., 2012; Iarussi et al., 2015; Panozzo et al., 2013; Huang et al., 2012; Liu et al., 2011; Campen and Kobbelt, 2014). A simpler direct, all-at-once rounding typically shows poorer (e.g. in the case of quad or hex mesh generation (Bommes et al., 2013b; Liu et al., 2011)) or even inacceptable (e.g. in the case of cross field generation) results(Bommes et al., 2010).

We present a formulation that allows us to apply this principle to the problem of domain and map construction for polycube mapping. This is in contrast to the previous approaches in this class of methods, which can abstractly be interpreted as (sometimes implicitly) making use of all-at-once discretization. The principle of successive discretization furthermore allows us to design our method to be of *intrinsic* nature, independent of the embedding and global orientation of the input model (cf. Figure 1) – a property not offered by all but one existing polycube-based method.

2. Related Work

The general field of hexahedral mesh generation is very broad. We focus our attention here on previous work related specifically to the generation of interior-regular hexahedral meshes.

Grid-based. Probably the simplest approach for this purpose is based on voxelization (*overlay grid method*), intersecting the given model with a Cartesian grid (Schneiders, 1995; Zhu and Blacker, 2000; Wan et al., 2011; He et al., 2009), possibly with structural post-processing (Yu et al., 2014; Yang et al., 2019). Adaptive grids can also be used (Schneiders et al., 1996; Zhang and Bajaj, 2006); their transitions between levels of resolution lead to meshes with interior singularities though. A major disadvantage of the grid-based approach is the lack of *boundary-sensitivity*. The worst-shaped elements are found at the surface, while the interior is perfectly regular. Unfortunately, in many applications high element quality is particularly desired near the boundary (Blacker, 2000).

Graph-based. A more boundary-sensitive result can be achieved by basing the mesh generation not on an extrinsic grid, but a structural graph embedded in the model (such as a shape skeleton or Reeb graph) and an accompanying surface segmentation (such as a pants decomposition) (Liu et al., 2015; Usai et al., 2016; Livesu et al., 2016; Li et al., 2013; Lin et al., 2008). This approach is best-suited for shapes consisting of tubular parts, not necessarily appropriate for generic use. For the design of a proper graph of high quality some manual efforts might be required in some methods.

Deformation-based. To be able to more directly take the resulting polycube map quality (thus mesh quality) into account, it is advisable to actually work with the map when determining the domain structure. One can initialize this with the identity map, and then (incrementally) modify it to turn an object's image under the map into a polycube. This can also be interpreted as deforming the input model with the goal of making its boundary piecewise aligned with the coordinate planes.

Most methods of this type make the choice of which coordinate plane to align which part of the model's boundary to based on the model's embedding and orientation in space (Gregson et al., 2011; Livesu et al., 2013; Yu et al., 2014; Huang et al., 2014; Fu et al., 2016), possibly with interactive editing options (Li et al., 2021). These approaches are able to quite robustly generate hexahedral meshes, however, due to this reliance on extrinsic information, from a quality point of view they often don't reach the quality of manually designed meshes. For instance, a bent cuboid (in a U or C shape), will develop corners (forming irregular boundary vertices) in the deformation process, even though intrinsically the geometry does not call for any such faults – it could simply be un-bent (cf. Figure 1, the lower half of the model). It is typically along these irregular boundary edges and corners that the distortion of the polycube map is particularly bad (cf. Figs. 10, 12 in (Fu et al., 2016)).

A further limitation is due to the common use of a simple map, rather than a chart-atlas map (as detailed in Section 3), which restricts the class of meshes and limits the mesh quality that can be achieved on models of non-zero genus.

The only non-extrinsic (and chart-atlas based) approach in this category for interior-regular meshes so far was described by Fang et al. (2016). They use a non-convex, non-linear optimization objective. It thus

requires a good initialization to yield good results. The objective employed to obtain an initialization has an inherent dilemma: it combines a term that penalizes the occurrence of internal irregularities and a term that promotes boundary alignment. One ultimately needs both properties (no irregularities and full boundary alignment) to yield a provably good result, but in this formulation one can only strike a balance between both terms. As was shown, the behavior of the procedure thus is somewhat unpredictable: it can work extremely well on several highly complex models, but at the same time break down and fail on models as primitive as a sphere. We formulate the problem of finding a polycube domain together with a map onto it as a mixed integer problem instead. Its solution can be approximated by a series of *convex* problems, each of which can be solved to the global optimum. Thus, while being computationally more intensive, we are able to avoid potential breakdowns due to bad initializations.

3. Problem Formulation

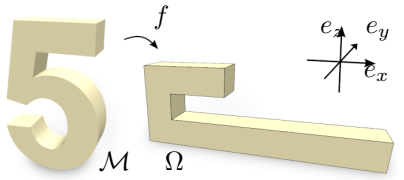
Consider a continuous, locally injective map f from a solid model $\mathcal{M} \subset \mathbb{R}^3$ to a 3-manifold domain Ω (with boundary):

$$f : \mathcal{M} \to \Omega.$$

In order to be a boundary-aligned, interior-regular map that implies a proper (infinitesimal) hexahedral mesh for $\mathcal{M}$, f needs to fulfil the following property:

$$((\nabla f)\, T_p\mathcal{M})^\perp \in \mathcal{E}, \quad \forall p \in \partial\mathcal{M} \backslash N \tag{1}$$

where $\mathcal{E} = \{e_x, -e_x, e_y, -e_y, e_z, -e_z\}$ is the set of oriented Euclidean coordinate axis vectors in $\mathbb{R}^3$. This means that the orthogonal complement of the mapped tangent space $T_p\mathcal{M}$ (i.e. the normal) is aligned with a coordinate axis. N is some network of curves on $\partial\mathcal{M}$ where this condition does not need to hold (corresponding to the edges of Ω, where ∇f and the normal are not defined – black in the inset). Quantization to a discrete grid is considered later in Section 8.



We use a discretization of $\mathcal{M}$ by means of a tetrahedralization, and consider ∇f piecewise constant, i.e. f linear per tetrahedron. The set of tetrahedra is denoted $\{t_i\}$ in the following. We assume each tetrahedron has at most one facet on the boundary $\partial\mathcal{M}$; this can easily be ensured by splitting. The boundary facet of tetrahedron t_i, if it exists, is denoted b_i, and its outward normal n_i.

$\mathcal{M}$ is given as input, f (the polycube map) and Ω (the polycube) are unknown and need to be constructed.

3.1. Chart-Atlas

The map f needs not be one continuous piece: we can use a chart atlas. Precisely, we can cut $\mathcal{M}$ into charts (for instance, each tetrahedron could be a separate chart), and require f to be related across charts by grid-preserving transition functions (composed of rotations from the chiral cubical symmetry group and translations) across the cuts (Nieser et al., 2011). By additionally requiring the transition functions to be cycle-consistent (their composition along any infinitesimal loop being the identity), f still implies a boundary-aligned interior-regular hexahedral mesh in $\mathcal{M}$. Without this additional identity property, the implied mesh would have irregularities (Nieser et al., 2011).

Cutting $\mathcal{M}$ into multiple charts actually does not yield any additional degrees of freedom in terms of obtainable mesh structures. However, if $\mathcal{M}$ is not simply-connected, of genus $g > 0$, cutting it into *one* simply-connected chart does yield the additional degrees of freedom that allow *any* interior-regular hexahedral mesh to be represented by f (see Fig. 3 and (Fang et al., 2016)). We give the necessary algorithmic steps for cutting in Section 7, after explaining the core method, for clarity, without consideration of cuts in the following.

3.2. Quality Objective

The parametric distortion incurred by the map f directly determines the element quality of the hexahedral mesh. We determine f using the following optimization.

We use a Poisson objective, constrained by (1):

$$f = \min_{f^*} \int_{\mathcal{M}} \|\nabla f^* - R_{\text{opt}}\|^2 dV, \tag{2a}$$

where $R_{\text{opt}} : \mathcal{M} \to \text{SO}(3)$ is a rotation field of minimal variation

$$R_{\text{opt}} = \min_{R} \int_{\mathcal{M}} \|\nabla R\|^2 \, dV, \tag{2b}$$

that itself is constrained to be axis-aligning via

$$Rn(p) \in \mathcal{E}, \quad \forall p \in \partial\mathcal{M}, \tag{1'}$$

where $n(p)$ is the surface normal at point p. Note that R defines a boundary-aligned orthonormal frame field $R^{-1}\mathcal{E}$ in $\mathcal{M}$ – an alternative perspective that may provide additional intuition. Also note how this constraint is effectively a weak version of (1).

So together we obtain f via the objective

$$f = \min_{f^*} \int_{\mathcal{M}} \left\| \nabla f^* - \min_{R} \int_{\mathcal{M}} \|\nabla R\|^2 \, dV \right\|^2 dV, \tag{2}$$

subject to constraints (1) and (1').

The choice of this Poisson objective, fitting the Jacobian of f to a maximally smooth axis-aligning rotation field, over other possible free-boundary mapping objectives such as ARAP (Liu et al., 2008; Chao et al., 2010; Alexa et al., 2000) or the recent one from Garanzha et al. (2021) was made not only because it is faster computationally, but because it proved to typically develop a lower number of irregular boundary edges and vertices (polycube corners) in surface regions that do not locally call for it.

3.3. Mixed-Integer Nature

This central problem (2) s.t. (1,1') is (or can be posed as) a mixed-integer problem. This is due to the discrete choices that need to be made in both types of constraints.

In detail, considering a discretized version of the problem, with ∇f and R piecewise constant, for every boundary facet b of $\mathcal{M}$ one out of six choices needs to be made: which of the six oriented axis directions $\pm e_x$, $\pm e_y$, $\pm e_z$ in $\mathbb{R}^3$ shall the normal of $f(b)$ be aligned to? As allowing different choices in (1) and in (1') is unreasonable, there is essentially one discrete degree of freedom per boundary facet.

4. Optimization Strategy

We start by considering the relaxed, non-integer form of the problem. This is simply (2) without the discrete constraints. Note that this is a convex problem. A global optimum can be obtained by first solving (2b) and then (2a); the latter, due to the 2-norm, comes down to a simple linear system solve. Note that the minimizer f of this relaxed problem is, up to a rigid transformation, the identity. We then proceed in two stages:

Stage I: Soft Alignment. A successive discretization strategy is applied to the relaxed problem to obtain a solution to (2) s.t. (1'). This process is along the lines of and in analogy to successive greedy integer rounding (Bommes et al., 2010). Effectively, a series of *continuous convex* optimizations is performed, starting with the relaxed problem and increasingly constrained by *convex* linear constraints (with *fixed* discrete degrees of freedom, e.g. "$= -e_y$" instead of "$\in \mathcal{E}$"). This is detailed in Section 5.

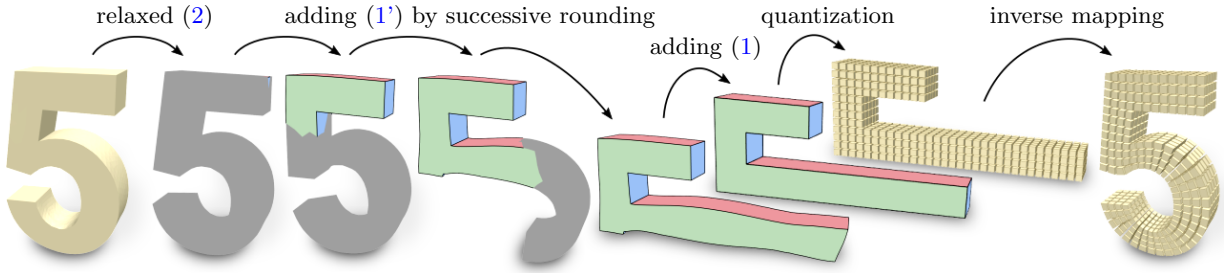


Figure 2: Demonstration of our method. An initial (identity) map and domain (gray) is obtained for the input model (left) via a relaxed continuous version of (2) s.t. (1'). The discrete degrees of freedom in (1') are then optimized by a successive mixed integer rounding strategy (the axis choices are visualized by corresponding colors). The exact constraints (1) are added afterwards to obtain a proper polycube domain. This is then quantized to integer edge lengths, allowing for a trivial hexahedral mesh generation via the canonical integer grid in $\mathbb{R}^3$. Mapped by the inverse map this yields a boundary aligned, interior-regular hexahedral mesh for the model (right).

Stage II: Hard Alignment. Afterwards, the constraints (1) are added to the problem, with *fixed* discrete choices, inherited from those already made for (1') in the previous stage. The resulting problem is *continuous* and *convex*. This stage is detailed in Section 6.

Figure 2 gives an overview of the entire process, from input model to final interior-regular hexahedral mesh.

5. Stage I: Soft Alignment

Our goal in this stage is to solve (2) s.t. (1'). The discretized version of (1') simply reads $R_i n_i \in \mathcal{E}\ \forall b_i$. We start with a continuous relaxation of this problem, where these discrete constraints are effectively replaced by non-discrete counterparts $R_i n_i \in \mathbb{R}^3$ (which is obviously tautological and satisfied when simply neglecting these constraints).

We define the *distance* $n - \mathcal{E} = \min_{e\in\mathcal{E}} \|n-e\|$, and the *closest axis* $e(n) = \operatorname{argmin}_{e\in\mathcal{E}} \|n-e\|$. With these definitions in place, we can apply a greedy, successive rounding strategy: the relaxed constraint $R_i n_i \in \mathbb{R}^3$ belonging to the boundary facet b_i for which $f(n_i) - \mathcal{E}$ is globally minimal, is replaced by the actual (fixed) constraint $R_i n_i = e(f(n_i))$. Here $f(n_i)$ is defined as the normal of $f(b_i)$, i.e. a normal of the object deformed by the current state of f (initially the identity). Notice the analogy with greedy mixed integer rounding (Bommes et al., 2010): the value that is closest to one of its possible discrete value choices (in the current solution) is fixed to this closest choice.

Whenever constraints are replaced, the problem is resolved. This amounts to

- first optimizing the convex problem (2b) (cf. Section 5.1)
  with an increasing number of constraints $R_i n_i = e$ (cf. Section 5.2)
- then optimizing the convex problem (2a) (cf. Section 5.3), updating the map f.

As we want an intrinsic formulation, we do not make the choices based on alignment to extrinsic global coordinate axis, but relative to already rounded neighborhoods. The set of fixed boundary facets thus forms a connected region at any stage of the process. Figure 2 shows an example of this process. Note that any polycube can be rigidly embedded in an axis-aligned manner in $\mathbb{R}^3$ in 24 different ways (based on the cubical symmetry group). To settle this degree of freedom, we initially fix two neighboring boundary facets to two arbitrary different axes, thereby making one out of $6 \times 4 = 24$ choices without loss of generality. We make this choice of two *different* axes where it is geometrically most appropriate, i.e. for the two boundary facets adjacent to the edge whose dihedral angle is globally closest to $\pi/2$. These two facets essentially form the seed of the successive rounding process, which can be imagined as a front propagation.

Notice that the process is entirely intrinsic: The global orientation of the input model does not have any effect on the result; even further, this intrinsic nature enables the "unrolling" of model parts (see Figure 2

or the twisted bar in Figure 6 bottom center), i.e. the initial extrinsic normal of a surface point does not tell anything about its final alignment under the polycube map we construct. This is in contrast to the various previous extrinsic approaches discussed in Sec. 2.

One can perform the successive rounding one by one, or fix multiple boundary facets in each iteration for faster processing. We choose to, per iteration, fix all facets that are directly adjacent to previously fixed ones. This showed only small differences compared to individual fixing, while significantly speeding up the process. If $\partial\mathcal{M}$ has multiple components (when $\mathcal{M}$ contains voids), the voids' boundary facets are processed successively after the outer boundary is done.

5.1. Optimizing (2b)

We need to decide what kind of representation to use for the rotation field R, i.e., how to express the rotation R_i per tetrahedron. Some previous work has used a spherical harmonics representation (Huang et al., 2011), which, however, is invariant with respect to cubical symmetry and thus is oblivious to singularities. Hence it does not allow for immediate strict control over the required interior-regularity of the result during the optimization process (possibly causing failure, as reported by Fang et al. (2016)).

Quaternions. We instead use quaternions to represent R. A quaternion $q_i = (q^w, q^x, q^y, q^z)^T$ per tetrahedron t_i represents R_i (via normalization in case q_i is not a unit quaternion). Compared to an alternative rotation representation by means of matrices, this representation is more compact, and simpler (fewer) constraints are required in the following.

Under this setting, objective (2b) can be discretized as follows:

$$q_{\text{opt}} = \min_{q} \sum_{ij} w_{ij} \left\| q_i - q_j \right\|^2, \quad \|q_i\| = 1 \ \forall i,$$

where the sum is over all non-boundary tetrahedral facets, with adjacent tetrahedra i and j. Instead of the per-element unit constraints $\|q_i\| = 1$, we relax the problem to a global unit constraint $\|q\| = 1$ as follows:

$$q_{\text{opt}} = \min_{\|q\|=1} \sum_{ij} w_{ij} \left\| q_i - q_j \right\|^2.$$

This allows us to efficiently formulate and solve this problem as an eigenvalue problem (Boyd and Vandenberghe, 2004):

$$L\, q_{\text{opt}} = \lambda_0\, q_{\text{opt}}, \tag{3}$$

where λ_0 is the smallest eigenvalue of L, a discrete Laplacian operator for the tetrahedral mesh (we used uniform weights in our experiments). Using the inverse power method (Monahan, 2011), the eigenvector q_{opt} belonging to the smallest eigenvalue is easily and quite efficiently found. Note that this relaxation is similar to the one used for globally optimal direction fields on surfaces by Knöppel et al. (2013).

5.2. Adding Constraints (1')

For a boundary facet b_i we need to add a constraint (1'), i.e. $R_i n_i = e_i$: the rotation in the corresponding tetrahedron is fixed such that it aligns the facet's normal with coordinate axis e_i (which is chosen as the closest axis, $e_i = e(f(n_i))$, based on the current state of f).

Let's consider the case of e_x being the closest axis to $f(n_i)$. How can we constrain q_i to represent a rotation that aligns n_i with e_x? This can be achieved by choosing $q_i = q_{n_i \to e_x}$, the rotation that aligns n_i with e_x by rotating around the axis $n_i \times e_x$. But choosing this specific q_i is unnecessarily restrictive; there is a whole subspace of aligning rotations: one can choose $q_i = q_{n_i \to e_x} \cdot q_{e_x}(\alpha)$, where $q_{e_x}(\alpha)$ is a rotation around the axis e_x by any angle α, because no such rotation affects the alignment to e_x.

$$q_{e_x}(\alpha) = \left(\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}, 0, 0 \right)^T \tag{4}$$

Since quaternion multiplication is linear in each quaternion we can express the product as $q_i = Q\, q_{e_x}(\alpha)$, with $Q \in \mathbb{R}^{4\times4}$ expressing multiplication from the left with $q_{n_i \to e_x}$. Hence for normal alignment we end up with the following four constraints on q_i:

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

It suffices to impose the latter two of these four constraints. This is due to the fact that q_i is going to be normalized, and any unit quaternion fulfilling the latter two conditions, i.e. of the form $(q^w, q^x, 0, 0)$, is automatically of the required form $\left(\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}, 0, 0\right)$ for some α.

In summary, constraining the quaternion q_i as needed is identical to imposing the homogeneous linear system $A\, q_i = (0,0)^T$ with $A \in \mathbb{R}^{2\times4}$ being the lower block of Q^{-1}. Note that due to homogeneity the constraints are unique and not affected by the sign ambiguity of quaternions, i.e. $A\,(\pm q_i) = \vec{0}$.

Alignment to different axes of $\mathcal{E}$ is done analogously.

Optimizing for the smoothest rotation field that aligns (a subset of) boundary facets' normals with prescribed axes can thus be done by augmenting the quadratic form with penalty terms:

$$q_{\text{opt}} = \min_{\|q\|=1} \sum_{ij} w_{ij} \left\|q_i - q_j\right\|^2 + \omega \frac{n_i}{n_b} \sum_{i\in\partial\mathcal{M}} \left\|A_i q_i\right\|^2,$$

with penalty factor ω, normalized by the numbers n_i and n_b of interior and boundary facets, respectively. Due to homogeneity of the penalty terms the optimality conditions of the resulting optimization problem are still an eigenvalue problem of the form (3) and can be efficiently solved to global optimality. Regarding the choice of ω, see Figure 8.

Optionally, in each iteration, we can reconsider the axis choice of facets already constrained in previous iterations, updating a constraint when another axis is actually closer. While this is very rarely the case, it occasionally helps to cure individual outlier facets. We use this option in our experiments.

5.3. Optimizing (2a)

Once we have obtained R_i (read from the normalized quaternions q_i) per tet using the quaternion field optimization, we can consider the discretization of (2a):

$$f = \min_{f^*} \sum_i \text{vol}_i \|\nabla_i f^* - R_i\|^2. \tag{5}$$

The minimum is obtained by solving a Poisson equation system:

$$Lf = G^T V R \tag{6}$$

with the Laplacian $L = G^T V G$ composed of the diagonal matrix of volumes, $V = \text{diag}(\text{vol}_\text{i})$, and the matrix G encoding the gradient operator for piecewise-linear functions on a tetrahedral mesh. As this system is invariant to global translations of f, to obtain a unique solution one vertex needs to be pinned down arbitrarily, or – even simpler – a Tikhonov regularizer can be added (adding a small multiple of the identity to the system matrix L; we use $10^{-6}\times$ its average eigenvalue, i.e. $\frac{1}{n}\text{trace}(L)$).

Note that R_i appears on the right-hand side only. Therefore the system can be prefactored once, and then solved by simple back-substitution every time the rotation field changes (i.e., everytime further constraints are added by successive rounding).

6. Stage II: Hard Alignment

After stage I, we have a domain (the image of $\mathcal{M}$ under the current f) that loosely resembles a polycube domain (cf. Figure 2 center bottom); its boundary does not yet fully comply with constraints (1), though.

In a second stage, we therefore add these hard alignment constraints, as detailed next. Before adding these, we apply the clean-up operations described by Gregson et al. (2011) to correct some typical per-facet

axis choice configurations that are not compatible with a polycube domain or are geometrically unfavorable. This includes merging patches (i.e. regions with constant axis choice) that have less than three neighbor patches with a neighbor patch by changing their axis choice, and altering the axis choice inside strips emanating from extremal turns of patch boundaries. In contrast to Gregson et al. (2011), we take cuts and transitions into account in this process to properly handle models of higher genus (cf. Sec. 7). Success of this clean-up procedure is not guaranteed; this is a general open challenge not specific to our method, as discussed further in Sec. 10. If this procedure changes some axis assignments, the constrained rotation field is re-solved before proceeding.

Note that alternatively, due to the intrinsic "pre-deformation" that stage I delivers, one could actually take the obtained domain as input for any existing *extrinsic* polycube generation method, e.g. (He et al., 2009; Gregson et al., 2011; Wan et al., 2011; Livesu et al., 2013; Yu et al., 2014; Huang et al., 2014; Fu et al., 2016; Yang et al., 2019), and obtain a final polycube domain of "intrinsic quality". The handling of cuts and transitions (cf. Sec. 7) would pose additional challenges for such an approach, though.

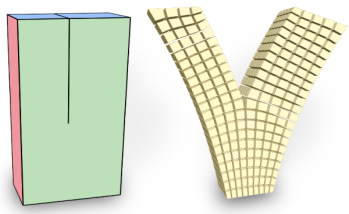
6.1. Adding Constraints (1)

We setup constraints (1) with fixed discrete choices, inheriting those made for (1') in the previous stage, i.e. normal n_i of boundary facet b_i is supposed to be aligned with the oriented axis e_i chosen in stage I (cf. Section 5.2).

In the discrete setting, constraint (1) can be expressed per boundary facet by requiring the facet to lie in one of the coordinate planes. Let (u_i, v_i, w_i) be the vertices of boundary facet b_i, which is supposed to have normal-alignment with oriented axis e_i. We require:

$$\begin{aligned} e_i^T(u_i - v_i) &= 0 \\ e_i^T(u_i - w_i) &= 0 \end{aligned} \qquad \forall b_i \tag{7}$$

Furthermore, we need to require the polycube's edges to be axis-aligned. In most cases this is implied by the above constraints, namely when the boundary facets on both sides of the edge are aligned to different (unoriented) coordinate axes. This is because non-coplanar axis-aligned planes intersect in an axis-aligned line. If both sides are aligned to the same axis e (e.g. to e_x and $-e_x$ respectively, see the slit in the inset for an example), however, we need to additionally require for the edge (u_j, v_j):



$$e_j^T(u_j - v_j) = 0 \qquad \forall j \in J \tag{8}$$

where e_j is the axis different from e and most perpendicular to $(u_i - v_i)$, and J is the set of all tetrahedral mesh edges that lie on polycube edges and require such an additional constraint due to the neighboring facets having same-axis alignment.

All these are linear equality constraints. Hence, the problem (6) s.t. (7) and (8) is continuous and convex, thus easily solved (cf. Section 8 for more details).

7. Higher Genus

As outlined in Section 3.1, we should virtually cut $\mathcal{M}$ into one simply-connected chart and allow transitions across the cuts, so as to enable full flexibility in cases where the genus of $\mathcal{M}$ is non-zero (cf. Fig. 3).

One previous approach considered cutting into a simply-connected chart (Fang et al., 2016). As stated by the authors, their employed cutting strategy is limited; planar cuts are used, which are too restrictive for complex geometries. In the following we describe a method for this task that is reliable on general shapes, at the expense of commonly somewhat larger cuts.

7.1. Determining Cuts

We start by computing a spanning tree in the dual graph of the tetrahedral mesh $\mathcal{M}$, rooted at an arbitrary tetrahedron. Then we (preliminarily) mark all interior facets whose dual edge is not part of the spanning tree as *cut facets*. Obviously, these cut $\mathcal{M}$ into a topological ball. Cut facets which are adjacent to an interior edge that is not adjacent to any other cut facet are then unmarked, as this does not change the simply-connectedness. Intuitively, dangling parts of the cut surface are retracted. This is done iteratively until no more facets can be unmarked. The remaining marked facets are the final cut facets. Figure 4 illustrates the cut facets on two examples. Interior edges adjacent to a cut facet are called cut edges.

7.2. Transition Functions

Across a cut facet c_{ij} between tetrahedra t_i and t_j the map f is related by a rigid transition function ϕ_{ij} (with rotational and translational component), i.e.

$$f_j = \phi_{ij} f_i \quad \text{on } c_{ij}, \tag{9}$$

where f_i and f_j is the (linear) restriction of f to tetrahedra t_i and t_j. For interior-regularity, these transitions need to be cycle-consistent, i.e. need to compose to the identity around any interior tetrahedral mesh edge:

$$\phi_1 \circ \cdots \circ \phi_m = I, \tag{10}$$

where the transitions ϕ_i are the oriented transitions across cut facets incident to a cut edge.

Most (often even all) interior cut edges have two adjacent cut facets. This implies that these facets simply need to have the same transition ϕ – because they are involved in opposite orientation, ϕ and ϕ^{-1}, in the composition $\phi^{-1} \circ \phi = I$ around the edge. If all cut edges have two adjacent cut facets this simply means that each connected component of cut facets has a constant transition ϕ. Note that there are g connected cut components in this case, where g is the genus of the mesh.

7.3. Determining Transitions

These transition constraints are taken into account as follows: at the end of stage I, when all discrete degrees of freedom of (1') have been fixed, we successively round the rotational component of the transition ϕ across each cut facet to one of the 24 rotations from the cubical symmetry group. This cannot be done in a trivial, greedy manner because we need to make sure that (10) remains feasible.

A cut edge incident to $m > 2$ cut facets is called a *complex cut edge*. A maximal connected component of cut facets not involving a complex cut edge is called a cut patch, cf. Figure 4. Pick a complex cut edge, determine the optimal rotation for one incident cut patch. Repeat this for all but one cut patches incident at the edge. The rotation of the last incident cut patch is implied via (10) as $\phi_m = (\phi_1 \circ \cdots \circ \phi_{m-1})^{-1}$. Pick a complex edge incident to one cut patch whose transition rotation is already determined and repeat this

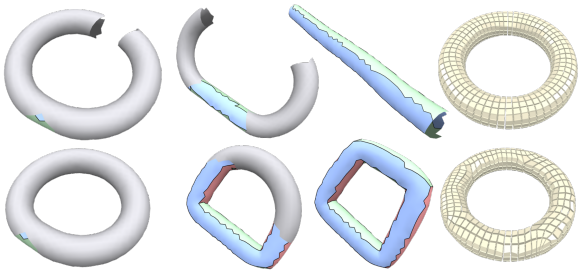


Figure 3: Illustration of the benefit of using a chart-based map definition. Top: with cuts and transitions, the torus can unroll, implying a hexahedral mesh without any singular corners. Bottom: without cuts and transitions, 16 corners emerge inevitably.

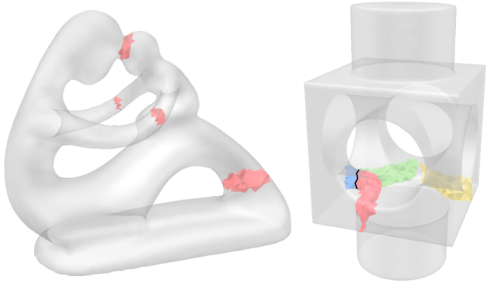


Figure 4: Illustration of cut facets inside two (semi-opaque) example models. In the left model there are four isolated cut patches. In the right model, there is one isolated cut patch (yellow) and three cut patches (red, blue, green) meeting at a sequence of complex cut edges (black).

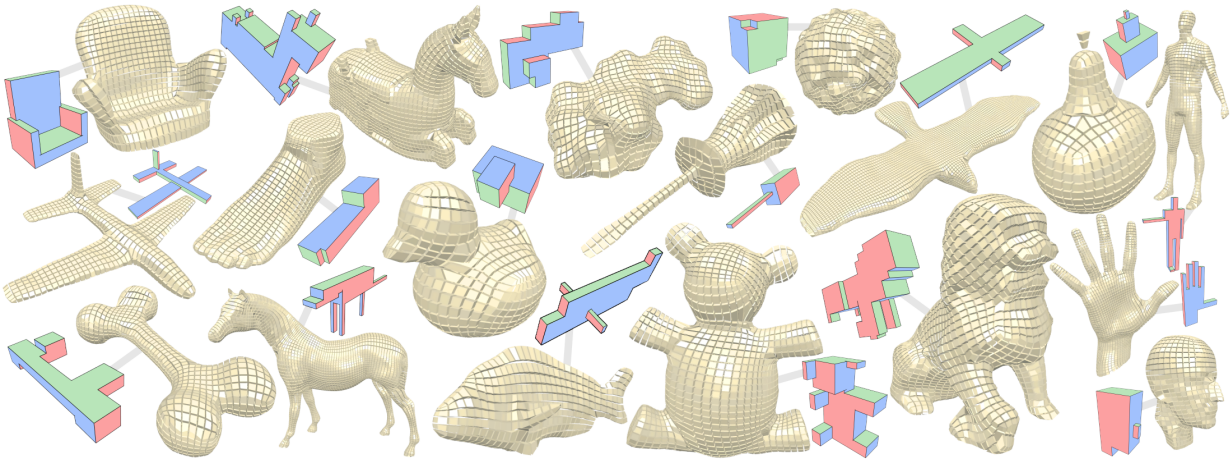


Figure 5: Example results on models of genus 0. On these relatively simple models the intrinsic approach's ability to "unroll" curved parts does not play a major role; its insensitivity to the extrinsic orientation of the input, however, does show its benefit.

until no further complex cut edge has incident cut patches with undetermined transition rotations. As each component of the cut surfaces is simply-connected, this greedy procedure does not lead to any inconsistencies violating (10) by construction. For each remaining cut patch (those not incident to any complex cut edge), simply determine the optimal rotation independently.

The translational components of ϕ are arbitrary, but subject to quantization, as described in the following.

8. Quantization

All variables that appear in constraints (7) or (8), as well as the translation variables of the cut transitions ϕ need to be integral (or from $s\mathbb{Z}$ for a scale factor s that controls the sizing) to make the polycube domain align properly with the regular cube tessellation of $\mathbb{R}^3$.

To this end, we again employ a greedy mixed integer rounding strategy (Bommes et al., 2010) to solve (6) s.t. (7), (8), (9), (10). This rounding effectively *quantizes* the polycuboid domain to a proper polycube domain. At a significantly higher computational cost, a Branch-and-Bound mixed integer solver could be applied as well for improved result quality; the actual benefit of course would strongly depend on the chosen target mesh resolution, for high-resolution meshes it can be minor.

While there are several specialized quantization approaches for polycubes (Cherchi et al., 2016; Protais et al., 2020; Zhao et al., 2019) (with potential benefits in efficiency) note that they are not applicable to non-zero genus domains with chart-atlas maps.

Finally, we apply the local injectivity promoting optimization from Garanzha et al. (2021) to reduce occasional local inversions in the final map f, and can then extract the hexahedral mesh implied by this polycube map, e.g. using the method of Lyon et al. (2016).

9. Results

We apply the intrinsic mixed-integer polycube method to a variety of models, of trivial topology or higher genus, of technical or organic nature, of primitive or complex shape. For each case in the following we show the resulting polycube domain $f(\mathcal{M})$ and an implied hexahedral mesh. We apply no postprocess optimization, simplification, straightening, padding, or smoothing, but show the immediately resulting hexahedral meshes so as to provide a raw insight without obscuring clarity by mixing the effects of multiple algorithms.

Examples. Figure 5 shows examples of relatively simple genus 0 models. These are also handled well by previous polycube generation and mapping approaches. Importantly, though, note that even for such models

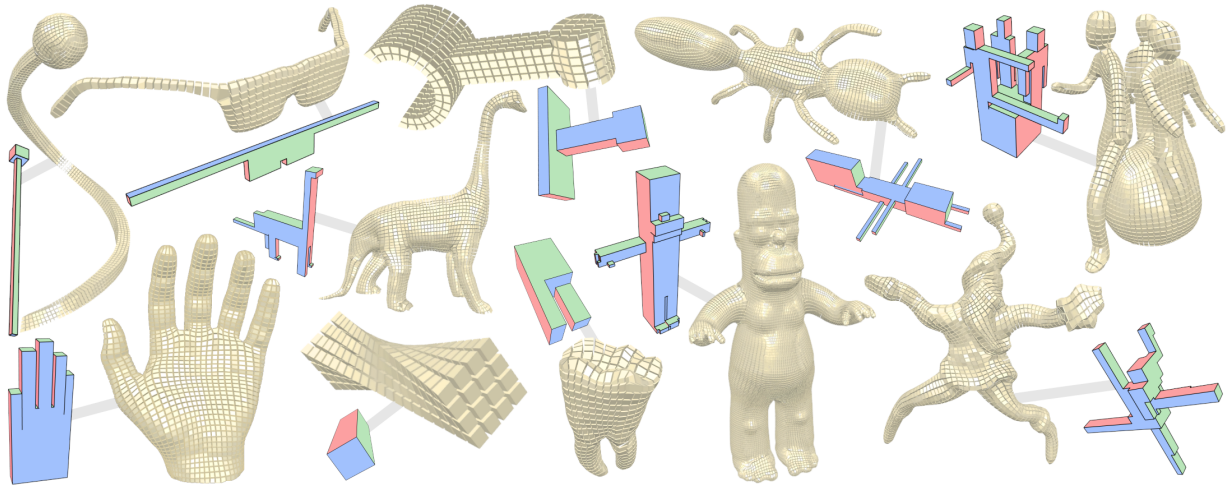


Figure 6: Example results on more complex models of genus 0. On these the intrinsic approach's "unrolling" nature shows its strength: spurious corners and edges in the polycube not called for by the intrinsic geometry (merely by the extrinsic embedding) are avoided in numerous places.

a benefit over extrinsic methods is that the result and its quality does not depend on the orientation of the input model; no matter how the input is rotated in space, exactly the same result is obtained.

Figure 6 shows examples of models (genus 0) where the intrinsic nature of our approach shows its strength (beyond input orientation independence) and makes a major difference. Extrinsic approaches would create undesirable spurious corners and edges, inducing additional irregular vertices and edges in the hex mesh in places where they are not geometrically called for, causing unnecessary distortion. Our intrinsic approach effectively "unrolls" and "untwists" certain parts of the models, avoiding such artificial irregularities.

Figure 7 shows examples of models with nontrivial topology. Here our use of a chart-atlas map (i.e., cuts with grid-preserving transitions, cf. Sec. 7) in combination with the intrinsic approach yields obvious benefits, providing more degrees of freedom in terms of resulting mesh structure, while still remaining in the space of interior-regular hexahedral meshes.

Parameters. In Figure 8 the effect of the choice of boundary alignment penalty ω (Sec. 5.2) is illustrated. A very low value commonly causes unnecessary corners and edges to emerge. An overly high value can cause small but desirable corners and edges to be missed, i.e. flattened out. The range $[3, 30]$ can be considered reasonable according to our experiments; we use a default of 10 as it provides a good general balance.

In Sec. 5 we mentioned how the "seed" of the successive process is chosen (at the globally most right-angled edge). One may wonder how sensitive the process is to the choice of seed. Figure 9 illustrates this on two examples. As can be seen, on the more technical model the resulting polycubes are structurally identical for different seeds. On the more organic model, where the ideal polycube structure indeed is not as evident, results differ structurally in some details, without being unreasonable. So, as could be expected, there is some dependence, i.e. different results could be obtained by using other seeds than our proposed choice, but we do not consider this a particularly purposeful or effective parameter.

Comparison. Above, we focused on a qualitative demonstration of the features of the proposed approach. A quantitative comparison to other approaches could provide further insight, but actually is an intricate matter. First of all, there is the question of what measures to actually look at. The number of unnecessary polycube corners seems attractive, but necessity is hard to formalize. When, more indirectly, looking at map distortion or implied hex mesh quality, results depend strongly on the employed post-processing (map distortion optimization, hex shape optimization, untangling, padding, etc.), the chosen hex mesh resolution, and the chosen map or mesh distortion measures. An insightful comparison would require an identical or close choice of these circumstances. Unfortunately, results reported in previous works on polycube map generation stem from largely varying setups, posing a challenge in this regard. In particular, differing

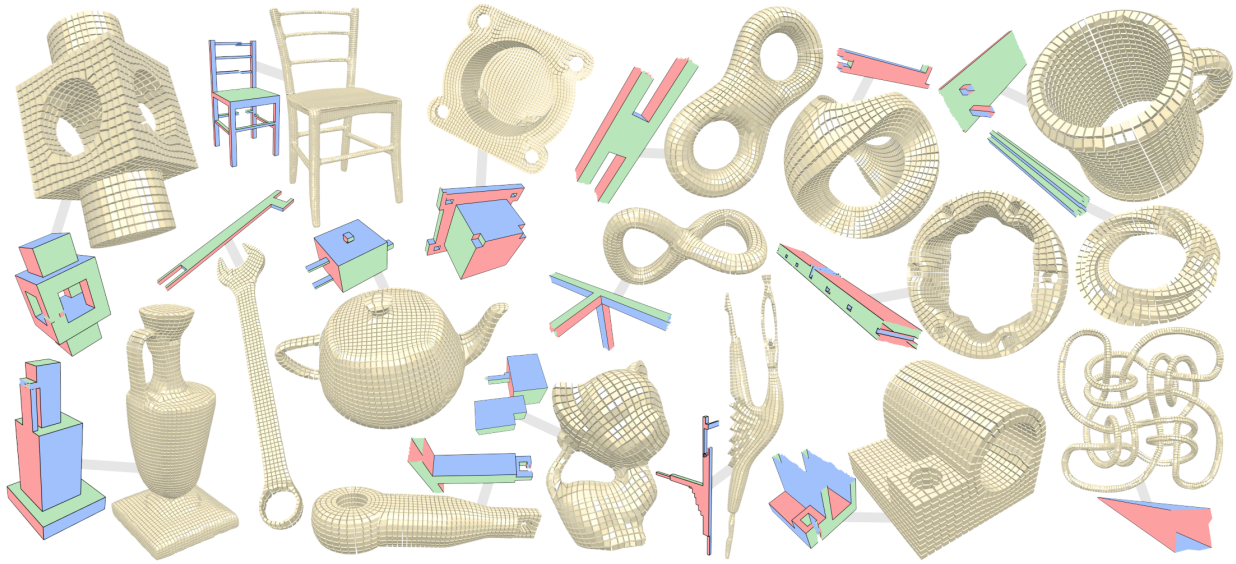
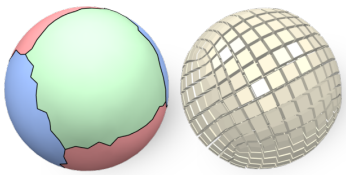


Figure 7: Example results on models of higher genus. Here the beneficial effect of our method's chart-atlas setting in synergy with its intrinsic nature can be observed clearly in many cases. Notice in particular the unrolling of cylindrical, toroidal, and knotted pieces. The cuts/transitions are visible as open interfaces in the polycube visualizations.

post-processes are employed (with or without map smoothing, with or without hex mesh smoothing, with or without hex mesh padding/buffering), not rarely using external tools and not always spelling out all relevant settings.

We therefore focus on the qualitative demonstration, and made our selection of models (Figs. 5 to 7) such that there is overlap with multiple previous works in terms of shown examples, enabling the reader to get a sense of the differences by visual comparison. With the above caveats in mind, we furthermore show the distribution of various quality measures evaluated on the shown hexahedral meshes in unoptimized, unpadded form in Figure 13, and make these raw meshes available as basis for future comparisons in varying optimization and postprocessing setups.



As a concrete point of comparison to Fang et al. (2016) (the one previous method that, like ours, is truly intrinsic) the inset shows our method's result on a sphere model – a reported hard failure case of this previous method, which in contrast to our method starts from a frame field with interior singularities and does not always (regardless of the seeming simplicity of the model) succeed in getting rid of them.

The proposed method is not particularly cheap computationally. Problems (2b) and (2a) need to be solved repeatedly. However, note that the latter can be pre-factored once and then efficiently be reused. As an example, using a 2018 commodity laptop, on the rightmost model in Figure 6 with 50K tetrahedra, stage

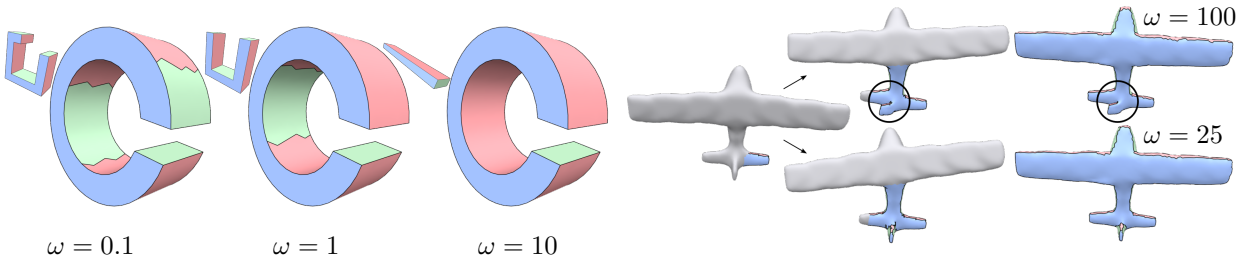


Figure 8: Effect of the choice of alignment penalty ω. Small values ($\ll 10$) cause spurious kinks in the polycube, whereas overly large values ($\gg 10$) may flatten details (like the plane's vertical rudder for $\omega = 100$), especially when they smoothly transition into their surround.

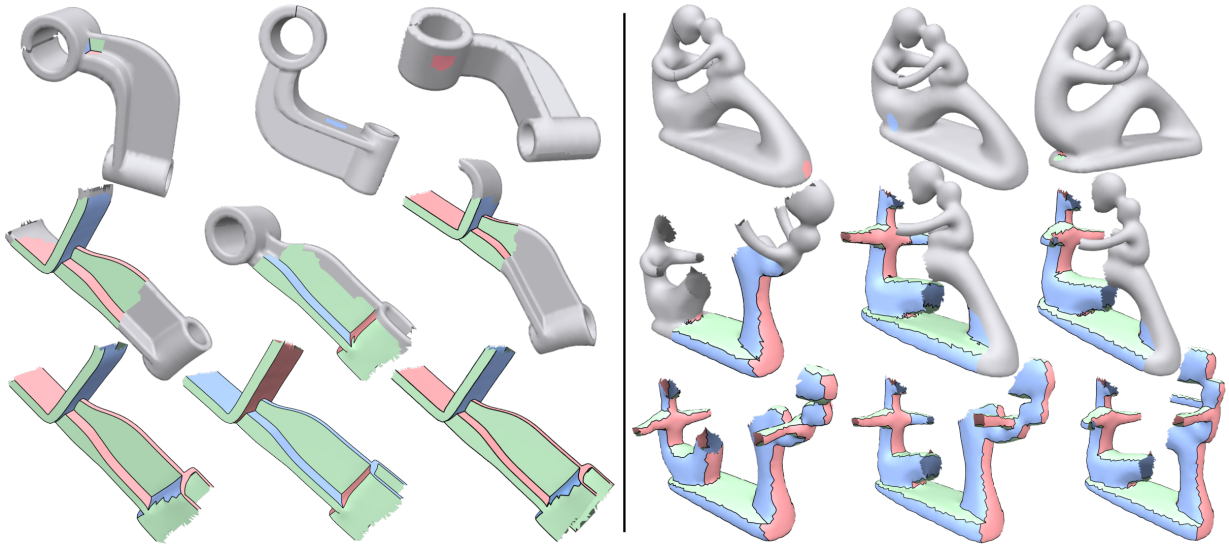


Figure 9: Effect of the choice of seed. Each column shows three snapshots of stage I, top to bottom: early after the start, in the middle, and at the very end. Gaps at handles in this visualization are due to the topology-induced cuts (Sec. 7).

I takes 418s (99.7% thereof spent in solving for the quaternion field q_{opt}), stage II (including quantization) takes 4.7s. Note that, in comparison, the previous intrinsic method of Fang et al. (2016) is reported to have taken in the range of 10 to over 30 minutes on meshes of this size.

Certainly, the progressive nature of the approach, while beneficial on the one hand, can have less favorable effects as well. Figure 10 shows examples where the result, while valid, has corners and edges in clearly suboptimal places in terms of geometric adequacy. In cases where these arise, they commonly appear towards the end of the process of successively adding alignment constraints, in the regions constrained last. While structurally the results are favorable in many cases, geometrically they are biased to some extent by the approach's progressive, greedy nature. It is an interesting avenue for future work to explore ways to combine the general successive rounding principle with a more balanced procedural strategy, while retaining all the benefits we observe. Note that hard feature curves (as opposed to weak or smooth features, Figure 11) are commonly well respected even though no explicit measures are taken to that end.

10. Future Challenges

There are some major challenges for the future in the polycube-based hexahedral meshing context, which our method as well as all related work is still facing. We discuss three key points in the following.

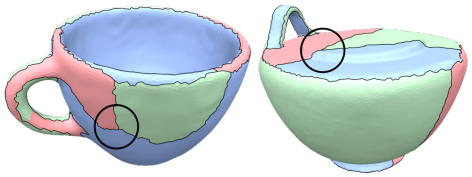


Figure 10: Two examples of models (of "polycube-unfriendly" shape) where, towards the end of the successive alignment process, corners end up into geometrically unfavorable near-flat places.

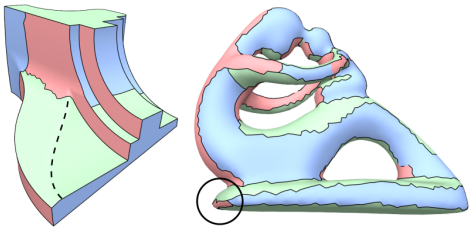


Figure 11: The method commonly yields patches well aligned with sharp features, as can be seen in Figures 5–9. Weak feature curves (left, dashed) are not aligned to naturally. At smooth (non-sharp) features, patch borders are not always optimally centered on the feature (right, circled).

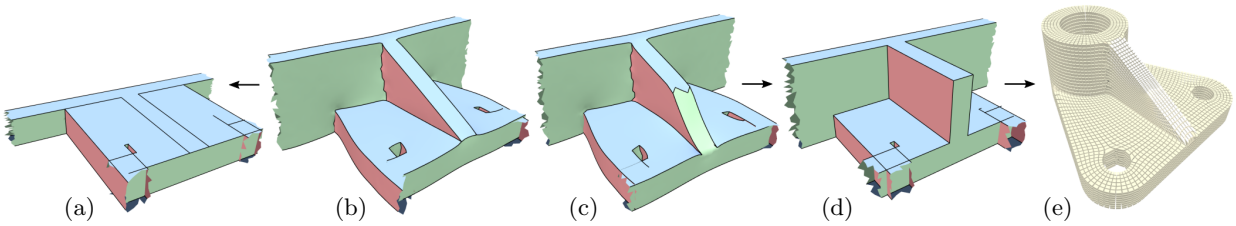
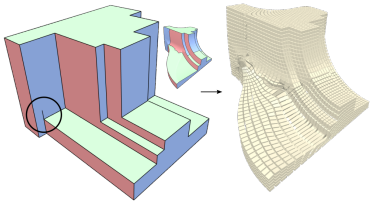


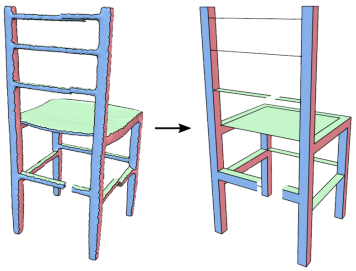
Figure 12: Example result (b) after stage I where the label repair rules from Gregson et al. (2011) are insufficient: the red triangular region is forced into degeneration under hard alignment constraints in stage II (a). Using a small manual label modification (c), a valid stage II result (d) is obtainable, implying a hexahedral mesh (e).

Alignment Validity. No suitable generic necessary and sufficient conditions on the $\mathcal{E}$-labeling (i.e. axis assignment) of surface facets being compatible with a polycube are known, and no algorithm that can produce polycube-compatible $\mathcal{E}$-labelings in full generality is known. An incompatible $\mathcal{E}$-labeling implies that the problem in stage II ((6) s.t. (7), (8), (9), (10)) is infeasible (if we require local injectivity of the map f) or necessarily leads to degeneracies in the result. Also the employed repair rules from Gregson et al. (2011) are not always able to handle all situations adequately; Figure 12 shows one example. While fault-tolerant mesh extraction procedures (Lyon et al., 2016) can gracefully deal with such cases in certain situations, there is no guarantee of success. Previous work has often made use of the graph-based polycube conditions presented by Eppstein and Mumford (2010). It was overlooked that these are formulated for graphs, whereas an $\mathcal{E}$-labeling constitutes an *embedded* graph (a graph with fixed edge ordering around nodes), for which they are neither sufficient nor necessary; Sokolov and Ray (2015) demonstrated this issue in a report. They proposed an alternative label adjustment procedure. This alternative, however, is unfortunately not applicable to chart-atlas maps in cases of non-zero genus. This is due to its reliance on the separability of the three dimensions – which are often intertwined by the chart transitions.

Map Injectivity. Guaranteeing (local) injectivity of a volumetric map such as f is a long-standing problem. No automatic volumetric polycube mapping approach so far can provide full guarantees in this regard. One can prevent inversions during the deformation process (Fu et al., 2016), but then convergence to a state that complies with all constraints is at stake. The static tessellation of the underlying tetrahedral mesh can be a major obstacle as well. Like all previous work, we thus effectively make use of a best effort approach. The inset shows an example where the result of stage II has an inversion in the circled area; while a structurally valid hex mesh can still be extracted in this example, it is very distorted at this spot. Finding a guaranteed injective volumetric mapping technique is a key challenge for future work. Maybe the robust volumetric *cube* map approach of Campen et al. (2016) can be generalized to *polycube* maps, or recent promising optimization techniques (Garanzha et al., 2021) can be extended to provide strict guarantees.



Robust Quantization. The problem of quantizing the domain, such that it aligns properly with the integer cube tessellation of $\mathbb{R}^3$ without locally degenerating or inverting, is challenging as well – in particular when coarse hexahedral meshes are desired: In the inset example some of the chair's rails and stretchers degenerate in the vertical direction under rounding to a too coarse target resolution. Note that the gaps are due to chart transitions (Section 7) and not a quantization artifact. Robust solutions to the 2D version of the quantization problem have been presented (Campen et al., 2015; Lyon et al., 2019, 2021), generalization to 3D is not straightforward but potentially promising. Specialized polycube quantization techniques (Cherchi et al., 2016; Protais et al., 2020) so far target the transition-free case only; generalizing them to the chart-atlas setting will be interesting.



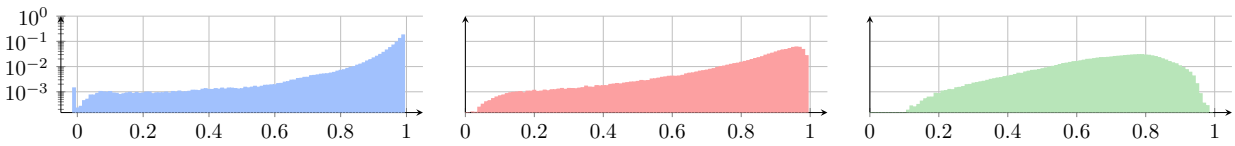


Figure 13: Log-scale histograms for • scaled Jacobian, • shape, and • stretch measures (Stimpson et al., 2007), collected over the elements of all hex meshes from Figures 5–7. The raw result meshes are considered, without any postprocess optimization, smoothing, or untangling. The relative amount of hexes with negative scaled Jacobian values is around 0.1% in these.

References

Alexa, M., Cohen-Or, D., Levin, D., 2000. As-rigid-as-possible shape interpolation, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co.. pp. 157–164.

Benzley, S.E., Perry, E., Merkley, K., Clark, B., Sjaardema, G., 1995. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis, in: Proc. 4th International Meshing Roundtable, pp. 179–191.

Blacker, T., 2000. Meeting the challenge for automated conformal hexahedral meshing, in: Proc. 9th International Meshing Roundtable, pp. 11–20.

Bommes, D., Campen, M., Ebke, H.C., Alliez, P., Kobbelt, L., 2013a. Integer-grid maps for reliable quad meshing. ACM Trans. Graph. 32, 98:1–98:12.

Bommes, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M., Zorin, D., 2013b. Quad-mesh generation and processing: A survey. Computer Graphics Forum 32, 51–76.

Bommes, D., Zimmer, H., Kobbelt, L., 2009. Mixed-integer quadrangulation. ACM Trans. Graph. 28, 77.

Bommes, D., Zimmer, H., Kobbelt, L., 2010. Practical mixed-integer optimization for geometry processing, in: Curves and Surfaces. Springer, pp. 193–206.

Bourdin, X., Trosseille, X., Petit, P., Beillas, P., 2007. Comparison of tetrahedral and hexahedral meshes for organ finite element modeling: an application to kidney impact, in: Proc. 20th Int. Technical Conference on the Enhanced Safety of Vehicles, Lyon, France.

Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press, Cambridge, England.

Campen, M., Bommes, D., Kobbelt, L., 2015. Quantized global parametrization. ACM Trans. Graph. 34, 192:1 – 192:12.

Campen, M., Kobbelt, L., 2014. Quad layout embedding via aligned parameterization. Computer Graphics Forum 33, 69–81.

Campen, M., Silva, C.T., Zorin, D., 2016. Bijective maps from simplicial foliations. ACM Trans. Graph. 35, 74:1–74:15.

Chao, I., Pinkall, U., Sanan, P., Schröder, P., 2010. A simple geometric model for elastic deformations. ACM Trans. Graph. 29, 38:1–38:6.

Cherchi, G., Livesu, M., Scateni, R., 2016. Polycube simplification for coarse layouts of surfaces and volumes. Computer Graphics Forum 35, 11–20.

Cifuentes, A., Kalbag, A., 1992. A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. Finite Elements in Analysis and Design 12, 313 – 318.

Eppstein, D., Mumford, E., 2010. Steinitz theorems for orthogonal polyhedra, in: Proc. 26th Annual Symposium on Computational Geometry, pp. 429–438.

Fang, X., Xu, W., Bao, H., Huang, J., 2016. All-hex meshing using closed-form induced polycube. ACM Trans. Graph. 35, 124:1–124:9.

Fu, X., Bai, C., Liu, Y., 2016. Efficient volumetric polycube-map construction. Computer Graphics Forum 35.

Garanzha, V., Kaporin, I., Kudryavtseva, L., Protais, F., Ray, N., Sokolov, D., 2021. Foldover-free maps in 50 lines of code. ACM Trans. Graph. 40, 102:1–102:16.

Gregson, J., Sheffer, A., Zhang, E., 2011. All-hex mesh generation via volumetric polycube deformation. Computer Graphics Forum 30, 1407–1416.

He, Y., Wang, H., Fu, C.W., Qin, H., 2009. A divide-and-conquer approach for automatic polycube map construction. Computers & Graphics 33, 369 – 380.

Huang, J., Jiang, T., Shi, Z., Tong, Y., Bao, H., Desbrun, M., 2014. ℓ1-based construction of polycube maps from complex shapes. ACM Trans. Graph. 33, 25:1–25:11.

Huang, J., Jiang, T., Wang, Y., Tong, Y., Bao, H., 2012. Automatic frame field guided hexahedral mesh generation. Technical Report. Tech. report, State Key Lab of CAD & CG, College of Computer Science at Zhejiang University.

Huang, J., Tong, Y., Wei, H., Bao, H., 2011. Boundary aligned smooth 3d cross-frame field. ACM Trans. Graph. 30, 143:1–143:8.

Iarussi, E., Bommes, D., Bousseau, A., 2015. Bendfields: Regularized curvature fields from rough concept sketches. ACM Trans. Graph. 34, 24:1–24:16.

Jiang, T., Huang, J., Wang, Y., Tong, Y., Bao, H., 2014. Frame field singularity correction for automatic hexahedralization. IEEE Transactions on Visualization and Computer Graphics 20, 1189–1199.

Kälberer, F., Nieser, M., Polthier, K., 2007. Quadcover – surface parameterization using branched coverings. Computer Graphics Forum 26, 375–384.

Knöppel, F., Crane, K., Pinkall, U., Schröder, P., 2013. Globally optimal direction fields. ACM Trans. Graph. 32, 59:1–59:10.

Li, B., Li, X., Wang, K., Qin, H., 2013. Surface mesh to volumetric spline conversion with generalized polycubes. IEEE Transactions on Visualization and Computer Graphics 19, 1539–1551.

Li, L., Zhang, P., Smirnov, D., Abulnaga, S.M., Solomon, J., 2021. Interactive all-hex meshing via cuboid decomposition. ACM Trans. Graph 40, 256:1–256:17.
Li, Y., Liu, Y., Xu, W., Wang, W., Guo, B., 2012. All-hex meshing using singularity-restricted field. ACM Trans. Graph 31, 177:1–177:11.
Lin, J., Jin, X., Fan, Z., Wang, C.C.L., 2008. Automatic polycube-maps, in: Chen, F., Jüttler, B. (Eds.), Advances in Geometric Modeling and Processing, Springer Berlin Heidelberg. pp. 3–16.
Liu, H., Zhang, P., Chien, E., Solomon, J., Bommes, D., 2018. Singularity-constrained octahedral fields for hexahedral meshing. ACM Trans. Graph. 37, 93:1–93:17.
Liu, L., Zhang, L., Xu, Y., Gotsman, C., Gortler, S.J., 2008. A local/global approach to mesh parameterization, in: Computer Graphics Forum, pp. 1495–1504.
Liu, L., Zhang, Y., Liu, Y., Wang, W., 2015. Feature-preserving T-mesh construction using skeleton-based polycubes. Computer-Aided Design 58.
Liu, Y., Xu, W., Wang, J., Zhu, L., Guo, B., Chen, F., Wang, G., 2011. General planar quadrilateral mesh design using conjugate direction field. ACM Trans. Graph. 30, 140:1–140:10.
Livesu, M., Muntoni, A., Puppo, E., Scateni, R., 2016. Skeleton-driven adaptive hexahedral meshing of tubular shapes. Computer Graphics Forum 35, 237–246.
Livesu, M., Vining, N., Sheffer, A., Gregson, J., Scateni, R., 2013. Polycut: monotone graph-cuts for polycube base-complex construction. ACM Trans. Graph. 32, 171:1–171:12.
Lyon, M., Bommes, D., Kobbelt, L., 2016. HexEx: Robust hexahedral mesh extraction. ACM Trans. Graph. 35, 123:1–123:11.
Lyon, M., Campen, M., Bommes, D., Kobbelt, L., 2019. Parametrization quantization with free boundaries for trimmed quad meshing. ACM Trans. Graph. 38, 51:1–51:14.
Lyon, M., Campen, M., Kobbelt, L., 2021. Quad layouts via constrained t-mesh quantization. Comp. Graph. Forum 40.
Monahan, J.F., 2011. Numerical Methods of Statistics. 2nd ed., Cambridge University Press, New York, NY, USA.
Nieser, M., Reitebuch, U., Polthier, K., 2011. CubeCover–parameterization of 3D volumes. Computer Graphics Forum 30.
Panozzo, D., Block, P., Sorkine-Hornung, O., 2013. Designing unreinforced masonry models. ACM Trans. Graph. 32, 91:1–91:12.
Protais, F., Reberol, M., Ray, N., Corman, E., Ledoux, F., Sokolov, D., 2020. Robust Quantization for Polycube Maps. Preprint.
Ramos, A., Simões, J., 2006. Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur. Medical Engineering & Physics 28, 916 – 924.
Reberol, M., Chemin, A., Remacle, J.F., 2019. Multiple approaches to frame field correction for CAD models, in: Proc. 28th International Meshing Roundtable.
Sarrate, J., Ruiz, E., Roca, X., 2014. Unstructured and semi-structured hexahedral mesh generation methods. Computational Technology Reviews 10, 35–64.
Schneider, T., Hu, Y., Gao, X., Dumas, J., Zorin, D., Panozzo, D., 2019. A large scale comparison of tetrahedral and hexahedral elements for finite element analysis. arXiv preprint arXiv:1903.09332 .
Schneiders, R., 1995. Automatic generation of hexahedral finite element meshes, in: Proc. 4th International Meshing Roundtable, pp. 103–114.
Schneiders, R., Schindler, R., Weiler, F., 1996. Octree-based generation of hexahedral element meshes, in: Proceedings of International Meshing Roundtable, pp. 205–215.
Sokolov, D., Ray, N., 2015. Fixing normal constraints for generation of polycubes. Research Report. LORIA.
Stimpson, C., Ernst, C., Knupp, P., Pébay, P., Thompson, D., 2007. The verdict library reference manual. Sandia National Laboratories Technical Report 9.
Tadepalli, S.C., Erdemir, A., Cavanagh, P.R., 2011. Comparison of hexahedral and tetrahedral elements in finite element analysis of the foot and footwear. Journal of Biomechanics 44, 2337 – 2343.
Usai, F., Livesu, M., Puppo, E., Tarini, M., Scateni, R., 2016. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. ACM Trans. Graph. 35, 6:1–6:13.
Vaxman, A., Campen, M., Diamanti, O., Panozzo, D., Bommes, D., Hildebrandt, K., Ben-Chen, M., 2016. Directional field synthesis, design, and processing. Comp. Graph. Forum 35.
Wan, S., Yin, Z., Zhang, K., Zhang, H., Li, X., 2011. A topology-preserving optimization algorithm for polycube mapping. Computers & Graphics 35, 639 – 649.
Wang, W., Zhang, Y., Liu, L., Hughes, T.J., 2013. Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology. Computer-Aided Design 45, 351 – 360.
Wei, X., Zhang, Y.J., Toshniwal, D., Speleers, H., Li, X., Manni, C., Evans, J.A., Hughes, T.J., 2018. Blended b-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis. Computer Methods in Applied Mechanics and Engineering 341, 609–639.
Yang, Y., Fu, X.M., Liu, L., 2019. Computing surface polycube-maps by constrained voxelization. Computer Graphics Forum 38, 299–309.
Yu, W., Zhang, K., Wan, S., Li, X., 2014. Optimizing polycube domain construction for hexahedral remeshing. Computer-Aided Design 46, 58–68.
Zhang, Y., Bajaj, C., 2006. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Computer methods in applied mechanics and engineering 195, 942–960.
Zhao, H., Li, X., Wang, W., Wang, X., Wang, S., Lei, N., Gu, X., 2019. Polycube shape space. Computer Graphics Forum 38, 311–322.
Zhu, J., Blacker, T., 2000. Overcoming cartesian grid generation obstacles, in: Proc. 7th Int. Conference on Numerical Grid Generation in Computational Field Simulations.