CUBECOVER – Parameterization of 3D Volumes

M. Nieser, U. Reitebuch and K. Polthier

Freie Universität Berlin, Germany †

Abstract

Despite the success of quad-based 2D surface parameterization methods, effective parameterization algorithms for 3D volumes with cubes, i.e. hexahedral elements, are still missing. CUBECOVER *is a first approach which provides both, a consistent theoretical framework for volume parameterization plus a full pipeline for generating a hexahedral tessellation of a given volume with boundary aligned cubes which are guided by a frame field.*

The input of CUBECOVER *is a tetrahedral volume mesh. First, a frame field is designed with manual input from the designer. It guides the interior and boundary layout of the parameterization. Then, the parameterization and the hexahedral mesh are computed so as to align with the given frame field.*

CUBECOVER *has similarities to the* QUADCOVER *algorithm and extends it from 2D surfaces to 3D volumes. The paper also provides theoretical results for 3D hexahedral parameterizations and analyses topological properties of the appropriate function space.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations—Computational Geometry and Object Modeling

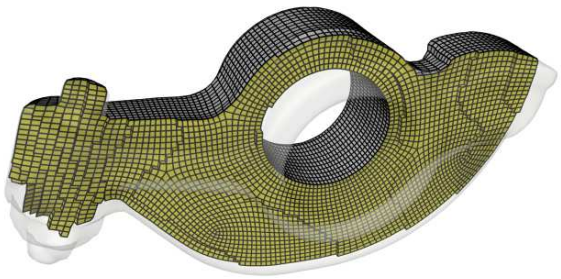

Figure 1: Section through the rockerarm volume. Hexahedral mesh computed with CUBECOVER.

1. Introduction

The field of surface parameterization was intensively researched during the recent years and various effective parameterization methods were invented. However, for general 3D volumes, boundary aligned parameterizations which are suited for hexahedral meshing remained out of reach.

† Supported by DFG Research Center MATHEON "Mathematics for key technologies"

CUBECOVER is a first approach to unfold a given 3D volume into $\mathbb{R}^3$ without fixing the boundary surface. Our approach computes a hexahedral parameterization from a given tetrahedral representation. Conceptually, CUBECOVER extends the surface parameterization algorithm QUADCOVER [KNP07] from surfaces to 3D volumes.

A clean and non-degenerate volume parameterization defines a hexahedral mesh suitable for PDE solving as well as a multilevel hierarchy of nested meshes. Such hexahedral meshes often enhance the speed and accuracy of PDE solvers significantly. Parameterizations can furthermore be used for volume sampling or 3D texture synthesis.

The presented approach analyzes the problem of volume parameterization from a theoretical viewpoint. A proper space of parameterizations with singularities is defined. For quadrangular parameterizations on surfaces, a singularity corresponds to an irregular vertex in the quad grid, i.e. with valence different than 4. In a volumetric hexahedral mesh, irregularities naturally occur at edges. An edge is considered as irregular if it is not incident to exactly 4 hexahedra. As a consequence, singularities in a volume parameterization form a network of curves throughout the volume. We provide a self-contained notion of volume parameterizations

with singularities and study topological properties of the underlying space.

Our approach is based on a guiding field which is used to locally steer the metric and the alignment of the parameterization. The pipeline to turn an input tetrahedral mesh into an output hexahedral mesh consists of three steps:

1. Design a *guiding frame-field*
2. Generate a *parameterization* which aligns to this field
3. Generate a *hexahedral mesh* from the parameterization

The first step produces a guiding field which consists of three vectors in each point of the volume forming a local coordinate frame. It encodes topological and metric properties and determines the shape of the parameterization. For surfaces, there are many different approaches about the design of vector- and frame-fields for parameterization. To our best knowledge, no comparable method exists for volumes. Designing fields in 3D is far more challenging than in 2D and it is not straightforward to generalize existing surface methods to volumes. It is out of the scope of this paper to give a fully automatic algorithm for field generation in 3D. Instead, we propose a simple design framework with user interaction. We expect much work on this topic in the near future.

The second step of our pipeline computes a parameterization which aligns to a given guiding field. CUBECOVER finds the optimal solution in L^2-sense; the gradients of the parameterization fit best possible to the input vectors. The algorithm is a generalization of the QUADCOVER algorithm for surfaces. However, the extension to volumes is not straightforward, since the space of parameterizations and the impact of singularities are far more complicated in 3D.

The third step of our pipeline is conceptually straightforward since extracting all iso-surfaces of a parameterization yields the faces of the hexahedral mesh. Nevertheless, the actual implementation is elaborate, but the challenges are mostly of technical nature (i.e. considering all occurring special cases) and are not the topic of this paper. Using a clear parameterization without any fold-overs guarantees that the output mesh is a pure hexahedral mesh, containing no other primitives.

In summary, our contributions are:

1. We settle theoretical foundations for describing volume parameterizations with singularities and analyze their properties.
2. We propose a framework for the manual design of volumetric guiding fields.
3. We introduce an algorithm which automatically computes a parameterization from a given guiding field.

1.1. Previous Work

Surface Parameterization. The field of 2D surface parameterization is extensively explored and already summarized in excellent surveys [FH05, SPR06]. Early work [HAT*00, GY03] studies conformal parameterization. The distortion of a parameterization can be reduced significantly by allowing cone singularities. Several methods place singularities in various ways and solve for a corresponding parameterization [TACSD06, RLL*06, KSS06, SSP08].

QUADCOVER [KNP07, KNP10] uses a user-defined input frame-field as guidance for the parameter lines. The parameterization problem is formulated as finding least squares fitting to the given field. The MIQ-algorithm [BZK09] is based on a similar formulation, but exchanges the rounding strategy of QUADCOVER by using a mixed integer solver. Additionally, a method for generating input frame-fields from a sparse set of vectors is proposed.

Volume Parameterization. Only few works exist on boundary-aligned volume parameterization. A first idea is to remesh the volume into hexahedra using a regular $\mathbb{Z}^3$ grid in the interior and adjust only the cubes along the boundary [ZB06, STSZ06]. Conversely, the Whisker-Weaving-algorithm [TBM95] starts with a quadrilateral mesh on the boundary and extends it to the inner volume. Another approach [SERB98] decomposes the volume into small volumetric patches using the embedded Voronoi graph of the object and then remesh the patches into hexahedra.

Mean value coordinates are used for 2D surface parameterization. These coordinates are extended for star-shaped polyhedral volumes in [FKR05]. [JSW05] proposed mean value coordinates for volumes bounded by arbitrary closed triangular meshes and applied it to texture mapping and volume morphing.

Harmonic functions are used to compute maps between two given volumes, as in [WGTY04] with applications to decomposing medical 3D data, and in [LGW*07, LGW*09] for deformation and volume morphing.

A hexahedral parameterization can also be obtained by mapping the volume to a polycube. There exist several approaches to compute polycube maps, but most of them only map the 2D surface onto a polycube surface. [HWFQ09, XHY*10] compute volumetric polycube maps which can be applied to hexahedral meshing. [LLWQ10] proposes the use of generalized polycube domains which may not have an embedding in $\mathbb{R}^3$. In the context of volume parameterization, polycube maps have the disadvantage that they may produce high metric distortion in the vicinity of edges and vertices of the polycube.

[MCK08] proposes a parameterization method for cylindric volumes applied for tri-variate spline fitting. [MC10] extends this approach to more general volumes using some kind of medial surface to decompose the volume into tensor-product patches. [XHH*10] uses Green's functions to map star-shaped volumes onto the unit sphere.

Most of the mentioned methods are cross-parameterization methods, i.e. they map the input volume to another pre-

defined volume, typically a simpler geometry such as the unit ball, a cylinder, a given polycube or any other volume. To our knowledge, there is no method which computes a parameterization from the input volume into $\mathbb{R}^3$ without prescribing the map at the boundary. Furthermore, most present methods compute parameterizations without any singularities. As known from 2D parameterizations, singularities are essential to reduce the overall distortion.

1.2. Approach and Paper Outline

We explore the natural space for parameterizations into $\mathbb{R}^3$ with singularities. The setting is used to compute such a parameterization similarly to QUADCOVER. However, it is also a canonical setting for other common parameterization approaches, e.g. using conformal functions. The setting is described in detail in Sect. 2.

Pipeline. The parameterization step of CUBECOVER takes a (bounded) 3D tetrahedral volume plus a guiding frame-field as input. The result is a hexahedral parameterization which aligns to these input frames and to the volume boundary. It consists of two main steps:

1. Compute a potential function of the frame-field.
2. Enforce integer conditions to obtain a globally continuous cubical mesh.

These two steps do agree with each other, i.e. they optimize the same energy. Similar to the pipeline of QUADCOVER, it does not matter if they are combined into one, looking for the best aligning globally continuous parameterization, or if they are solved successively.

Despite the similarities to QUADCOVER, it is not straightforward to extend the 2D setting to 3D. In 2D, the aim is to compute a parameterization which aligns to surface features, e.g. the principal curvature directions. Since the Euclidean three-dimensional space is flat, there are no canonical directions coming from the curvature tensor. Instead, we are given the volume boundary as additional constraint, where the cubes have to fit. Furthermore, the parameterization space is different. The point singularities from 2D extend to one-dimensional singular curves which meet in node points. These curves induce special constraints to the parameterization function which are not present in the 2D case.

Coverings. Similarly to QUADCOVER, the space of parameterizations is strongly related to branched covering spaces. In the volume case, a parameterization can be interpreted as a scalar-valued function on a 24-sheeted branched covering volume. This viewpoint helps to understand topological properties of parameterizations and frame-fields.

Although branched coverings provide a clear and self-contained theoretical framework, there is no need to explicitly compute the covering inside the algorithm. The topology of the covering can be fully described by the so-called *matching matrices* (see Sect. 2.2). Therefore, we simplify our presentation and describe CUBECOVER without the notion of covering spaces. The algorithm can be read and understood without any previous knowledge of QUADCOVER.

Field Design. We propose a method for inducing a frame-field from a manually designed *meta-mesh*. It is a very coarse hexahedral mesh which encloses the volume and roughly encodes desired topological and geometrical properties.

With using a meta-mesh, the designer has explicit control over the alignment of the hexagons in the inner. If the meta-mesh e.g. consists of a single hexahedron, then the parameterization tries to align everywhere to the edges of this hexahedron. More complex meta-meshes can be used to induce singularities and reduce overall distortion (see Sect. 3.2).

Designing a meta-mesh is much simpler than the tessellation of the volume itself. The meta-mesh may be very coarse, it may contain T-junctions and does not rely on having uniform hexahedra or alignment to the boundary. CUBECOVER globally optimizes the sizes of the latter hexahedra and their alignment to the meta-mesh, while fitting exactly to the boundary.

Note that using a meta-mesh for parameterization is fundamentally different to a polycube parameterization. Meta-meshes are more general and provide natural parameterizations for volumes with a different structure than a polycube (as in Fig. 3). Polycubes have no inner singularities. Instead, they induce boundary singularities at edges of the polycube which often leads to increasing distortion (as in Fig. 16).

The algorithm is described in detail in Sect. 3. In Sect. 4 we discuss results of parameterized volumes.

2. Setting

We now describe the underlying theoretical background, introduce data structures for volume geometry and frame-fields, and discuss singularities of frame-fields in hexahedral meshes. The concepts are fundamental for volume parameterizations in general.

2.1. Parameterization

Given an underlying volumetric geometry as subset $V \subset \mathbb{R}^3$, bounded by a two-dimensional closed surface $M = \partial V$. A *volume parameterization* of V is a map $f : V \to \mathbb{R}^3$, $p \mapsto (u,v,w)^T$.

A non-degenerate parameterization f induces a hexahedral tessellation in a natural way by $f^{-1}(C)$, where C is the regular cube tessellation of $\mathbb{R}^3$ (Fig. 2):

$$C := \{(x,y,z) \in \mathbb{R}^3 \,|\, x \in \mathbb{Z} \vee y \in \mathbb{Z} \vee z \in \mathbb{Z}\}. \qquad (1)$$

For aligning to the boundary surface, one coordinate has to be integer in each point on the boundary. This guarantees that the cubes align tangentially to the surface and induce a regular quad mesh on it.

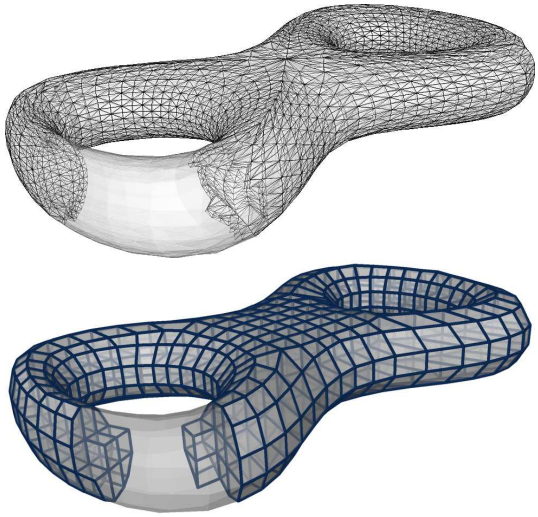

Figure 2: Tetrahedral input- and hexahedral output mesh.

We consider the volume as a 3-dimensional manifold with an atlas, where each tetrahedron (short: *tet*) is a single chart. Two adjacent charts (tets) overlap at their common face. A parameterization f is discretized as being linear on each tet, and is represented by its values at the four vertices. Given two adjacent tets s and t, the parameter function f may have different values in both charts. They are related by the *transition function* between s and t which we restrict to those positively oriented linear functions which leave the standard cube grid C invariant. Thus, even though f may have multiple values on faces of tets, the induced hexahedral grid has no visible seams (see Fig. 3). The transition is given as:

$$f_{|t} = \Pi_{st} f_{|s} + g_{st}, \tag{2}$$

where g_{st} is a constant vector in $\mathbb{Z}^3$ (called *gap between s and t*) and denotes an integer translation of the cube grid. Π_{st} (called *matching matrix* between s and t) is an element of the chiral cubical symmetry group, i.e. any map in $SO(3)$ which maps coordinate axes to coordinate axes. This group contains 24 different transformations.

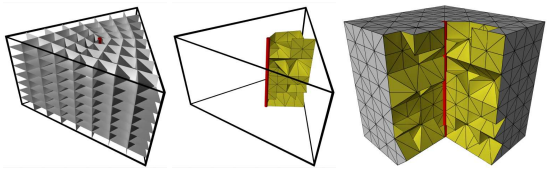

Figure 3: 3D parameterization of a triangular prism. Left: Induced hexahedral mesh. Middle: Faces where f is discontinuous. Right: Image of volume in texture space.

2.2. Frame-Fields and Singularities

The parameterization is guided by a so-called *frame-field*. It is defined by 3 three-dimensional vectors (U_t, V_t, W_t) in each tet t which are constant per tet and form a local coordinate system. Each parameterization $f = (u, v, w)^T$ defines a frame-field by its component-wise gradient fields $(\nabla u, \nabla v, \nabla w)$.

In adjacent charts s and t, the matching Π_{st} determines how the frames are combinatorially connected. The frame (U_t, V_t, W_t) in t is combinatorially identified to a rotated frame in s which can be described by the formal matrix-vector multiplication: $\Pi_{st} \cdot (U_s, V_s, W_s)^T$ (Fig. 4, left).

An important characteristic of a parameterization is the location and types of its *critical points* (or the *singularities* of the gradient frame-field). For classical vector fields, these are points where the Jacobian is singular. They are typically characterized by their *index*, which is an integer number describing the behavior of the field in the vicinity. On 2D manifolds, this notion was generalized to frame-fields [KNP07, NP09]. When used for quadrangular parameterization, the frame-field index at any point is always an integer multiple of $1/4$, typically $1/4$ or $-1/4$.

In three dimensions, singularities of non-degenerate frame-fields in general form one-dimensional curves throughout the volume (like the red curve in Fig. 3). We denote this network of singular curves as *singularity graph*. Singularities cannot simply start or end in the interior; they usually meet in *node points*, or they hit the bounding surface.

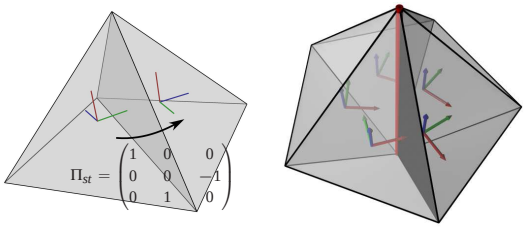

Figure 4: Left: Frame-field with matching between two tets. (U, V, W) (red, green, blue) is mapped to $(U, W, -V)$. Right: Frame-field in the vicinity of a singularity.

The classical definition of a vector field index is not sufficient to fully classify 3D frame-field singularities. We therefore introduce the *type* of a singularity which provides additional information in our discrete setting.

Given a volume V tessellated with tets, let e be an (oriented) edge of V and γ be a small closed loop (positively oriented) around e. Starting in an adjacent tet t_0, the loop passes through all tets $(t_0, \ldots, t_k, t_0)$ in the edge star. The concatenation of matchings between all passed charts

$$\text{type}(e, t_0) := \Pi_{t_k t_0} \circ \Pi_{t_{k-1} t_k} \circ \cdots \circ \Pi_{t_1 t_2} \circ \Pi_{t_0 t_1} \tag{3}$$

is called the type of the edge with respect to t_0 and is also a cubical symmetry transform. When tracking a frame along γ, the type measures how it is transformed when coming back to the start point. If the type is not the identity map Id, the edge is called *singular* (Fig. 4, right), otherwise it is a *regular edge*. Note that the type is independent of the reference tet t_0, but is just described in its local coordinate system, i.e. with respect to tet t_1, the type is

$$\text{type}(e,t_1) = \Pi_{t_0t_1} \circ \text{type}(e,t_0) \circ \Pi_{t_0t_1}^{-1} \tag{4}$$

which is just a basis transform of $\text{type}(e,t_0)$.

Theorem 2.1 Let f be a parameterization where no tet is mapped to a degenerated tet (a tet with vanishing volume). Then, the type of each edge is either the identity or a rotation around one of the coordinate axes.

Proof. Given an edge e with the (cyclically ordered) tets $(t_0,\dots,t_k)$ in the edge star and a point p on e. The parameterization f respects the transition functions (2), i.e.

$$\begin{aligned} f_{|t_1}(p) &= \Pi_{t_0t_1} f_{|t_0}(p) + g_{t_0t_1}, \\ f_{|t_2}(p) &= \Pi_{t_1t_2} f_{|t_1}(p) + g_{t_1t_2}, \quad \dots \end{aligned}$$

and therefore with plugging each equation into its successor we get:

$$\begin{aligned} & f_{|t_0}(p) = \text{type}(e,t_0) \cdot f_{|t_0}(p) + g \\ \Leftrightarrow\ & (\text{Id} - \text{type}(e,t_0))\, f_{|t_0}(p) = g \end{aligned} \tag{5}$$

for some constant vector $g \in \mathbb{Z}^3$ which depends on the gaps $g_{t_it_{i+1}}$. This equation is true for all points p on the edge. Therefore, if the matrix $(\text{Id} - \text{type}(e,t_0))$ has full rank, then the whole edge gets mapped to a single point (the solution for $f_{|t_0}(p)$ in Eqn. (5)) and all tets in the edge star would degenerate. The only edge types (from the chiral cubical symmetry group) where the matrix is not regular are the identity and rotations around a coordinate axis.

□

The theorem restricts the number of possible singularity types of a proper parameterization. Even though each individual matching can be any of the 24 cubical symmetries, the resulting edge types must be one of the ten suitable ones:

$$\{\text{Id}, J_u^k, J_v^k, J_w^k \mid k \in \{1,2,3\}\}, \tag{6}$$

where J_u, J_v and J_w are the 90 degree rotations about the respective coordinate axis.

As a consequence, a singularity of a volume parameterization is always similar to a 2D singularity but extruded along the third coordinate. Let e.g. e be a singular edge whose type J_w^k is a w-axis rotation. From Eqn. (5) then follows, that the u and v coordinates for f are constant on e and uniquely determined when all gaps are known, whereas the w coordinate is independent of the gaps and can vary along the edge.

2.3. Singularities in a Hexahedral Mesh

Having introduced singularities of a frame-field in Sect. 2.2, we now describe their effect on the final hexahedral mesh. This section shows some observations about 3D frame-fields which we think are worth mentioning. They are not essential for the implementation of the algorithm, but they help to understand some topological properties of hexahedral meshes.

Let e be an edge e.g. of type J_w^k, $k \in \{1,2,3\}$. Computing the inverse matrix in Eqn. (5), it follows that u and v are multiples of $1/2$ (since $g \in \mathbb{Z}^3$). In case of $u,v \in \mathbb{Z}$, the singularity is contained in an integer iso-plane for u and v, and therefore, the singularity is represented by an edge in the later hexahedral mesh. Otherwise, the singularity would pass through the interior of a cube, producing other primitives, e.g. a prism over a 5-gon (Fig. 5).

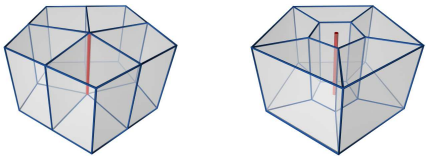

Figure 5: Valence 5 singularity. Left: The singularity stays on hexahedral edges. Right: Unintended situation, turning a hexahedron into a prism over a 5-gon.

Since we are interested in a pure hexahedral tessellation, we enforce $u,v \in \mathbb{Z}$. Singularities then appear as those edges in the hexahedral mesh with valence different from 4 (number of adjacent hexahedra). Similar to the 2D case, the valence of an edge in a hexahedral mesh is closely related to the type of the corresponding singularity.

The following theorem shows that singularities cannot be chosen arbitrarily. There is a relation between the valences of all edges incident to a vertex and therefore between singularities which meet in a node point.

Theorem 2.2 Let p be an inner vertex of a non-degenerated hexahedral mesh and e_i its adjacent edges. Then,

$$\sum_i (6 - \text{valence}(e_i)) = 12. \tag{7}$$

Proof. Consider an infinitesimal small sphere around vertex p. The hexahedral mesh induces a triangulation on that sphere where each hexahedron corresponds to a triangle, each two-dimensional face to an edge and each one-dimensional edge to a vertex of that triangulation (see Fig. 6). The proposition now becomes a formula for the vertex valences which is true for arbitrary triangulations of the 2-sphere and follows from the Euler formula.

□

An interesting observation about Eqn. (7) is that singularities of valence 6 do not contribute to the sum, but edges of valence 4 (regular edges) do. Fig. 7, left and right show an

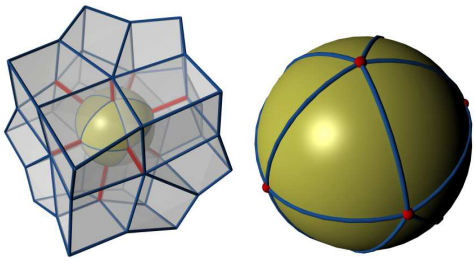

Figure 6: Infinitesimal sphere around a vertex of a hexahedral mesh with induced triangulation.

example where several singularities meet in one node point. The singularities in the right example are exactly the same, but with an additional valence 6 singularity.

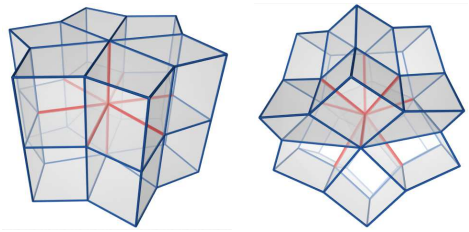

Figure 7: Left: 6 edges of valence 4 and 2 of valence 6 meet in a node point. Right: 6 edges of valence 4 and 3 of valence 6 meet in a node point.

Nevertheless, there cannot be a node point adjacent to 6 regular edges and only one valence 6 edge, even though it would fulfill Eqn. (7). A singularity cannot simply end somewhere in the volume. A singular edge adjacent to a vertex enforces that there is either another edge with the same index (the singularity passes "through" the vertex) or it splits up into two or more other singularities.

Given an arbitrary singularity graph, there is yet no sufficient condition on whether there exists a frame-field which adheres to these singularities. In 2D, satisfying the Poincaré-Hopf index theorem is enough to find a frame-field to given singularities. In 3D, more conditions are required to construct a correct singularity layout. The CUBECOVER algorithm does not rely on these conditions and will always find the best parameterization to the given input field. If the field is incompatible in the sense that there exists no valid potential function, then parts of the parameterization will degenerate. Our proposed frame-field generation method delivers a valid frame-field, since it is derived from a valid hexahedral mesh, the meta-mesh (see Sect. 3.2).

Behavior at the Boundary. At the boundary, a regular edge has valence 2. There may also be edges with valence 1 or ≥ 3 which we call *boundary singularities* of the volume parameterization. Boundary singularities are curves running on the boundary surface and should not be confused with singular points of the induced quad mesh.

A boundary singularity always corresponds to a change of boundary conditions, i.e. on the left and right side of the singular curve, a different coordinate u, v or w is snapped to the boundary.

3. Algorithm

3.1. Parameterization Guided by a Frame-Field

We now describe the CUBECOVER algorithm based on the setting of hexahedral parameterizations from Sect. 2. The output is a parameter map $f = (u,v,w)$ whose integer isosurfaces form a hexahedral tessellation of V. The user input to this parameterization step is a tet mesh V together with a frame-field X (matchings Π_{ij} plus frames (U,V,W)) and optionally conditions at the boundary. Boundary conditions enforce that the hexahedra align with one face to the boundary, i.e. one coordinate function is constrained to be constant along the boundary. The specification of boundary conditions also includes the information about which coordinate u, v or w is held fix in each triangle of the boundary surface. See Sect. 3.2 for details on our framework for generating input fields.

The parameterization is optimized for the best possible alignment of the parameter lines to the given frames, i.e. it minimizes the energy

$$E(f) = \int_V \|\nabla f - X\|^2 \mathrm{d}vol \tag{8}$$

The minimization is done in the space of piecewise linear maps f, i.e. given by its value $f_{|t}(p)$ at each vertex p of each tet $t \in V$. Note that a vertex can have different values in different tets. All unknowns are collected in the vector

$$\vec{f} = (u_0, v_0, w_0, \ldots, u_N, v_N, w_N)^T \in \mathbb{R}^{12 \cdot \#\text{tets}}$$

Derivation of Eqn. (8) by all unknowns leads to a linear system of equations $L\vec{f} = b$ with the Laplace matrix L and a right side b containing the discrete divergence of the frame-field.

Additional linear constraints are given by Eqn. (2) introducing the gaps g_{ij} as additional unknowns. Finding a global minimum of the energy is challenging since some variables are constrained to integers: the gaps $g_{ij} \in \mathbb{Z}^3$ and the snapped coordinate at the boundary must be in $\mathbb{Z}$. This problem of optimizing a quadratic functional under integer constraints is known as the *closest vector problem* which is NP hard, thus we cannot compute the optimal solution in reasonable computing time. Instead, we use the following heuristic for a nearly optimal solution: In the first stage, we ignore the integer constraints. In adjacent tets (s,t) and all three edges (p,q) of the common face we just enforce a constant gap, i.e.:

$$f_{|t}(p) - f_{|t}(q) = \Pi_{st}(f_{|s}(p) - f_{|s}(q)). \tag{9}$$

This system of linear constraints, together with given boundary conditions, can be written as $Cu = 0$, $C \in \mathbb{Z}^{\#\text{constraints} \times 12 \cdot \#\text{tets}}$. We optimize the energy using Lagrange multipliers by solving

$$\begin{pmatrix} L & C^t \\ C & 0 \end{pmatrix} \begin{pmatrix} \vec{f} \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{10}$$

with $\lambda \in \mathbb{R}^{\#\text{constraints}}$. The solution defines a parameterization f whose gaps can be computed from Eqn. (2).

The following observation helps to reduce the dimension of the system of equations drastically: The texture coordinates $f_{|t}$ of a single tet can be translated by a constant integer vector without violating any constraints or changing the energy. Therefore, there are many redundant variables in the system which can be eliminated by first computing a spanning tree T on all tets, and second, setting the gaps to 0 between tets which are adjacent in T.

In a second stage, the integer constraints are enforced. For all vertices on a singularity, there are two coordinate functions (e.g. u and v for a singularity of type J_w^k) which need to be integers (see Sect. 2.3). If boundary constraints are given, the corresponding coordinate function also has to be integer at the boundary. If we enforce these constraints, then it follows that the gaps are also integer since they are connected to the coordinate functions via Eqn. (2).

Let $u_{i_0}, \ldots, u_{i_n}$ be the integer variables in u. We successively round them to the nearest integer value and eliminate them from the system of equations (10) (which deletes the i_j-th row and column and modifies the right vector). Finally the system gets solved again with fixed u_{i_j}.

We solved the equations with a conjugate gradient solver. The solution of the first system is used as an initial start value for the second step. The final solution is then used for remeshing the volume into cubes by computing the intersection of each tet with the unit grid in texture space.

3.2. Frame-Field Construction

Generating good parameterizations requires suitable input frame-fields. In principle, an arbitrary field (e.g. constant frames) can always be used. It often makes sense to have specific control over the frame-field since it fully defines the location and type of singularities: inner singularities are given by the matchings Π_{st}, boundary singularities are introduced wherever the fixed coordinate at the boundary changes (see Sect. 2.3). Singularities always exist and, if badly placed, may induce high distortion. The automatic volumetric frame-field design is an unsolved issue as already mentioned and part of independent research as in the case of 2D surface parameterization.

Meta-Mesh. We propose a simple method for generating a frame-field from a *meta-mesh*. It consists of a (coarse) set of hexahedra with possible T-junctions and fully encloses the input volume V.

By mapping each hexahedron onto the unit cube, the standard orthonormal basis frame induces a trilinear frame-field in the actual hexahedron. The frame-field is evaluated at the barycenters of all inner tets providing a frame-field on the tetrahedral input mesh.

The matchings Π_{st} and therefore the singularities are also induced from the meta-mesh: Let s, t be adjacent tets which lie in the hexahedra c_s, resp. c_t. If $c_s = c_t$, then Π_{st} is set to identity. Otherwise, it is defined as transformation which maps the axes of c_s to those of c_t (in the image of the unit cube).

Finally, the boundary conditions are induced from the frame-field: In each boundary tet, the frame vector that is best aligned with the surface normal is used to constrain the corresponding coordinate function.

Relaxing. Since singularities always run along edges of the tet mesh, they are usually not smooth, but start to zigzag. In practice, this is no problem and CUBECOVER runs very stable even when the singularity curves are not smooth. However, the cubes of the output mesh stick to singularities, i.e. the edges of the cube mesh which align to singularities are still not smooth. We handle this issue by relaxing the mesh afterwards allowing the cubes to become more regular. Positions of all vertices are iteratively altered by adding a displacement vector. Each hexahedron defines a displacement vector for all of its vertices which deformes this hexahedron into a perfect cube. Per vertex, the displacement vectors are then averaged. After each iteration step, boundary vertices are projected back onto the boundary (Fig. 8).

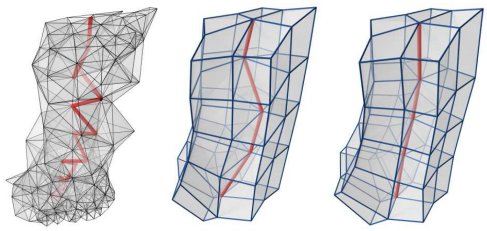

Figure 8: Left: Frame-field singularity in the input mesh runs along tet edges. Middle: Singularity in the cube mesh after parameterization with CUBECOVER. Right: Relaxed cubes with smoother singularity.

4. Results

We applied the CUBECOVER algorithm to various synthetic and general models. In all our experiments, the algorithm runs very stable and the results are regular and smooth. Skull and Hand model are provided by the Aim@Shape Repository. The bush model is courtesy of the authors of [ZB06].

Fig. 9 shows the parameterization of a torus. The frame-field was designed manually with two singularities. The red

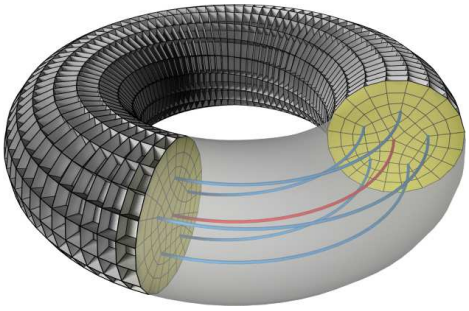

Figure 9: Twisted torus.

singularity runs exactly along the axis of the torus. The blue singularity circulates 5 times around the torus with a slight twist. The parameterization generates a regular hexahedral mesh. A slice of this mesh corresponds to a 2D parameterization of a solid 2-ball with 5 index $1/4$ singularities and 1 index $-1/4$ singularity.

The fandisk (Fig. 10) is computed using a constant frame-field pointing in direction of the coordinate axes. Notice the exact alignment of the parameterization to sharp features of the surface since they are exactly represented by boundary singularities.

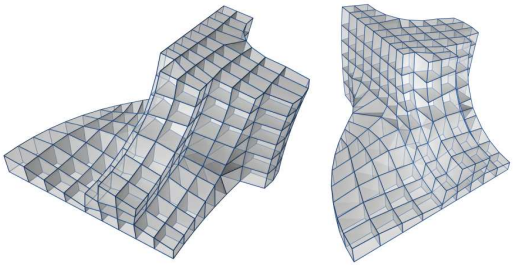

Figure 10: Feature alignment at the fandisk.

The rockerarm is shown in Fig. 1. The frame-field was constructed using 26 cubes as meta-mesh (Fig. 14, top right). The parameterization contains 5 singularities which emanate through the volume from the front side to the backside.

Fig. 11 shows the hexahedral meshes generated for artificial genus 4 and 5 volumes. For the genus 4 model, an axis parallel frame-field is used whereas the genus 5 model was parameterized with a simple meta-mesh containing 15 cubes (Fig. 14, bottom middle). Fig. 12 and 13 show further parameterizations of the Hand and the Skull model. The hand model shows that it is less important whether the meta-mesh is an approximation of the volume itself. Even if several fingers are represented by one meta-cube, the correct boundary singularities are induced on each individual finger.

Designing a Meta-Mesh. Creation of a meta-mesh is much simpler than producing a hexahedral tessellation of the input

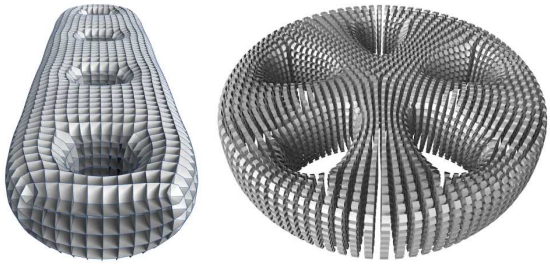

Figure 11: Volumetric parameterization of artificial shapes of genus 4 and 5.

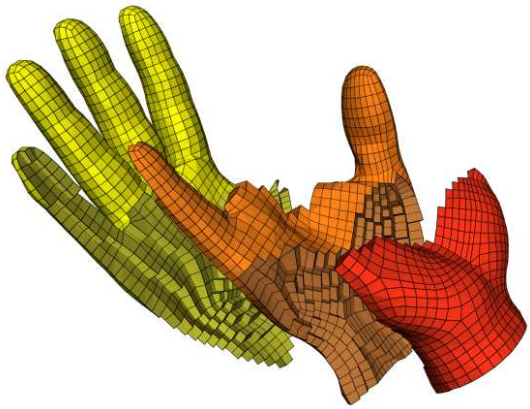

Figure 12: Digital hand model.

volume. The meta-mesh may be very coarse, may contain T-junctions and does not have to fulfill any boundary conditions. CUBECOVER then optimizes the hexahedra for equal cube sizes and alignment to the meta-mesh. E.g. in Fig. 15 (top), we re-parameterized the volume of Fig. 2 given a sin-

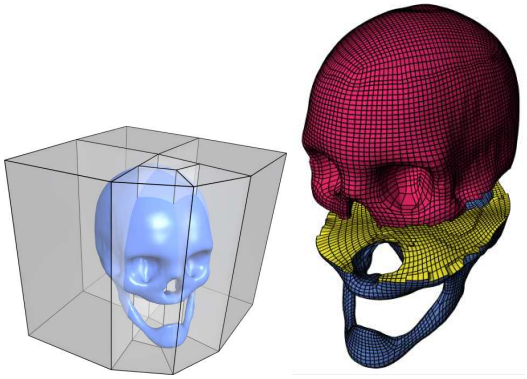

Figure 13: Parameterization of the skull. Left: Coarse cube meta-mesh used for defining a frame-field in the volume. Right: Remeshed hexahedral model.

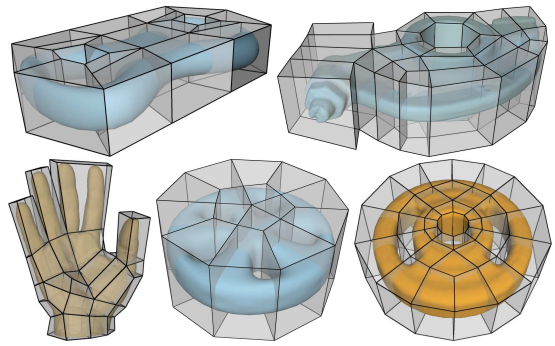

Figure 14: Meta-meshes for defining a frame-field. Top: Pretzel from Fig. 2, Rockerarm from Fig. 1. Bottom: Hand from Fig. 12, genus 5 model from Fig. 11, Bush from Fig. 17.

gle axis aligned cube as meta-mesh. Notice how the number of hexahedra between the two red stripes adapt locally (6 hexahedra in the middle part, 10 hexahedra at the ends) which is not inherited from the meta-mesh.

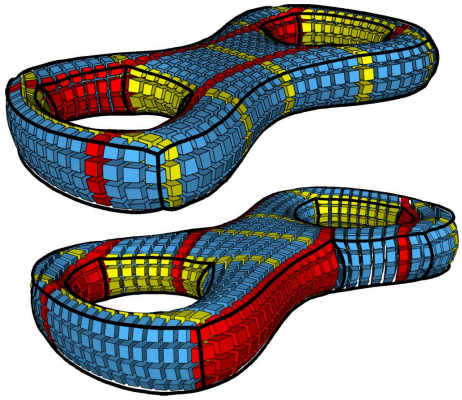

Figure 15: Model from Fig. 2, parameterized with constant frames, parallel to axes (top) and slightly twisted (bottom). Boundary singularities are drawn in black.

The most important task for designing a meta-mesh is to consider where to put singularities. In our tests, we observe an increasing distortion near singularities, similar to effects appearing in 2D quad parameterizations. Thus, singularities should be used sparsely and put at meaningful locations.

For comparing the influence of different meta-meshes, we parameterized the same volume with a slightly twisted constant frames (Fig. 15, bottom). Notice that both frame-fields induce boundary singularities which do not always fit to highly curved regions of the boundary surface. As a consequence, hexahedra in the vicinity of such singularities are distorted. More precisely, if a boundary singularity is placed where the boundary surface is almost planar, then it may force inner angles to become nearly 180 degrees (Fig. 16, left). This issue can be resolved by adapting the meta-mesh: If the boundary is mostly smooth and contains no sharp edges, all singularities should be offset into the inner of the volume (Fig. 16, right). The quality of angles is then increased much.

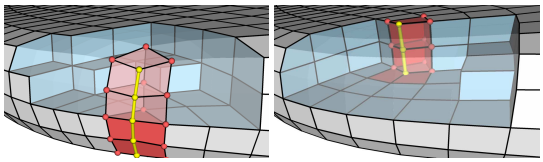

Figure 16: Boundary singularities can cause bad inner angles (left). Offsetting into the inner (right) enhance quality.

Comparison. The following table shows properties of the hexahedral mesh (number of hexahedra, the averages/standard deviation of dihedral angles, edge lengths, cube volumes).

Model	|H|	Angle	Length	Volume
Pretzel (Fig. 2)	528	90/17.4	.92/.26	.74/.32
Pretzel (15 top)	1256	90/14.8	.69/.11	.32/.09
Pretzel (15 bottom)	1452	90/16.3	.66/.12	.27/.08
Genus 4 (11)	11602	90/11.9	1.1/.17	1.4/.2
Genus 5 (11)	9459	90/9	1.8/.35	6.1/1.9
Torus (9)	7120	90/6.6	1.9/0.5	6.8/2.3
Fandisk (10)	268	90/9.65	4.2/1.0	76/29
Rockerarm (1)	35502	90/8.4	0.84/0.17	.58/.16
Hand (12)	4904	90/10.3	4.4/1.2	86/45
Skull (13)	40687	90/18.6	1.5/.4	3.0/1.5
Bush (17, [ZB06])	87885	90/12.2	.49/.09	.12/.03
Bush (17, Ours)	96054	90/5.2	.48/.07	.11/.02

Fig. 17 (left) shows a hexahedral mesh produced with the method from [ZB06]. The inner hexahedra are axis aligned and perfect cubes. Notice that the standard deviation of dihedral angles is lower in the mesh from CUBECOVER (right), since it distributes the distortion evenly in the volume.

For the time complexity, the algorithm has to solve two sparse linear systems of equations. We implemented CUBECOVER in Java and did not optimize it for performance yet.

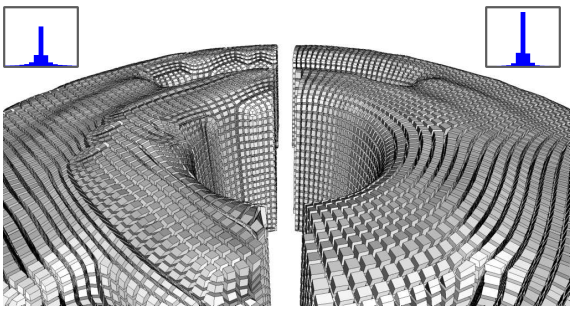

Figure 17: Bush model with [ZB06] (left) and CUBECOVER (right). The histograms show dihedral angles ranging from 60 to 120 degrees.

The time for setting up the matrix takes around 1 or 2 minutes. The main time is used for solving the system of equations the first time. The second system is smaller and uses the solution of the first one as start value in the conjugate gradient method. Thus, it is solved much faster. The following table shows the number of tets from the input mesh and the time used for setting up the matrix and solving the system (1st and 2nd time). Times were produced on a dual core processor with 2.8 GHz.

Model	|T|	Matrix	Solve 1	Solve 2
Pretzel (Fig. 2)	82944	54.7 s	57 s	3.7 s
Genus 4 (11)	27235	9.3 s	5.1 s	562 ms
Genus 5 (11)	36142	20.4 s	10 s	703 ms
Torus (9)	17280	55.2 s	3.3 s	266 ms
Fandisk (10)	23327	5.9 s	2.8 s	656 ms
Rockerarm (1)	55979	43.6 s	46 s	3.6 s
Hand (12)	125131	92.4 s	15:40 min	3.9 s
Skull (13)	156137	99.1 s	2:43 min	6 s
Bush (17)	79936	100 s	3:52 min	4.6 s

5. Conclusion and Future Work

We proposed the CubeCover algorithm to compute a 3D hexahedral mesh of a given volume. The hexahedra are aligned to a guiding frame-field and to the volume boundary. We also derived several theoretical conditions on the singularities and the gradient frame-field. Finally, we proposed a method for designing a frame-field using a manually generated meta-mesh.

We observed similar limitations known from 2D surface parameterizations methods: Badly placed singularities can lead to distortion. Flipped tets may appear if the distortion is too high. For now one can handle such situations by post-processing, or try to avoid by preprocessing the frame-field (e.g. smoothing).

The design of suitable frame-fields on surfaces is focus of ongoing research. Similarly, we expect intensive research on 3D frame-fields in the near future.

References

[BZK09] Bommes D., Zimmer H., Kobbelt L.: Mixed-integer quadrangulation. Siggraph 2009. 2

[FH05] Floater M. S., Hormann K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling* (2005), Springer, pp. 157–186. 2

[FKR05] Floater M. S., Kós G., Reimers M.: Mean value coordinates in 3d. *CAGD 22*, 7 (2005), 623–631. 2

[GY03] Gu X., Yau S. T.: Global conformal surface parameterization. In *SGP* (2003). 2

[HAT*00] Haker S., Angenent S., Tannenbaum A., Kikinis R., Sapiro G., Halle M.: Conformal surface parameterization for texture mapping. *TVCG 6*, 2 (2000), 181–189. 2

[HWFQ09] He Y., Wang H., Fu C.-W., Qin H.: A divide-and-conquer approach for automatic polycube map construction. *Comput. Graph. 33* (2009), 369–380. 2

[JSW05] Ju T., Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. Siggraph 2005. 2

[KNP07] Kälberer F., Nieser M., Polthier K.: QuadCover - surface parameterization using branched coverings. *Comp. Graph. Forum 26*, 3 (2007), 375–384. 1, 2, 4

[KNP10] Kälberer F., Nieser M., Polthier K.: Stripe parameterization of tubular surfaces. In *Topological Methods in Data Analysis and Visualization*. 2010. 2

[KSS06] Kharevych L., Springborn B., Schröder P.: Discrete conformal mappings via circle patterns. *Trans. Graph. 25*, 2 (2006). 2

[LGW*07] Li X., Guo X., Wang H., He Y., Gu X., Qin H.: Harmonic volumetric mapping for solid modeling applications. In *Solid and physical modeling* (2007), pp. 109–120. 2

[LGW*09] Li X., Guo X., Wang H., He Y., Gu X., Qin H.: Meshless harmonic volumetric mapping using fundamental solution methods. *Autom. Sci. Engin. 6* (2009), 409–422. 2

[LLWQ10] Li B., Li X., Wang K., Qin H.: Generalized polycube trivariate splines. SMI '10. 2

[MC10] Martin T., Cohen E.: Volumetric parameterization of complex objects by respecting multiple materials. *Comput. Graph. 34* (2010), 187–197. 2

[MCK08] Martin T., Cohen E., Kirby M.: Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *Solid and physical modeling* (2008), pp. 269–280. 2

[NP09] Nieser M., Polthier K.: Parameterizing singularities of positive integral index. In *Mathematics of Surfaces* (2009), pp. 265–277. 4

[RLL*06] Ray N., Li W. C., Lévy B., Sheffer A., Alliez P.: Periodic global parameterization. *Trans. Graph. 25*, 4 (2006), 1460–1485. 2

[SERB98] Sheffer A., Etzion M., Rappoport A., Bercovier M.: Hexahedral mesh generation using the embedded voronoi graph. In *7th International Meshing Roundtable* (1998), pp. 347–364. 2

[SPR06] Sheffer A., Praun E., Rose K.: Mesh parameterization methods and their applications. In *Foundations and Trends in Computer Graphics and Vision* (2006), vol. 2, pp. 105–171. 2

[SSP08] Springborn B., Schröder P., Pinkall U.: Conformal equivalence of triangle meshes. Siggraph 2008. 2

[STSZ06] Shepherd J. F., Tuttle C. J., Silva C. T., Zhang Y.: *Quality Improvement and Feature Capture in Hexahedral Meshes*. Tech. rep., SCI Institute Utah, 2006. 2

[TACSD06] Tong Y., Alliez P., Cohen-Steiner D., Desbrun M.: Designing quadrangulations with discrete harmonic forms. *SGP* (2006), 201–210. 2

[TBM95] Tautges T. J., Blacker T., Mitchell S. A.: The whisker weaving algorithm: A connectivity based method for constructing all-hexahedral finite element meshes, 1995. 2

[WGTY04] Wang Y., Gu X., Thompson P. M., Yau S. T.: 3d harmonic mapping and tetrahedral meshing of brain imaging data. In *Med. Imag. Comp. and Comp. Assist. Interv.* (2004). 2

[XHH*10] Xia J., He Y., Han S., Fu C. W., Luo F., Gu X.: Parameterization of star-shaped volumes using green's functions. In *Adv. Geom. Model. Proc.*, vol. 6130. 2010, pp. 219–235. 2

[XHY*10] Xia J., He Y., Yin X., Han S., Gu X.: Direct-product volumetric parameterization of handlebodies via harmonic fields. In *Int. Conf. Shape Model.* (2010), pp. 3–12. 2

[ZB06] Zhang Y., Bajaj C.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Comp. Meth. in Appl. Mech. and Engin. 195* (2006), 942–960. 2, 7, 9