Bijective Volumetric Mapping via Star Decomposition

STEFFEN HINDERINK, HENDRIK BRÜCKLER, and MARCEL CAMPEN, Osnabrück University, Germany

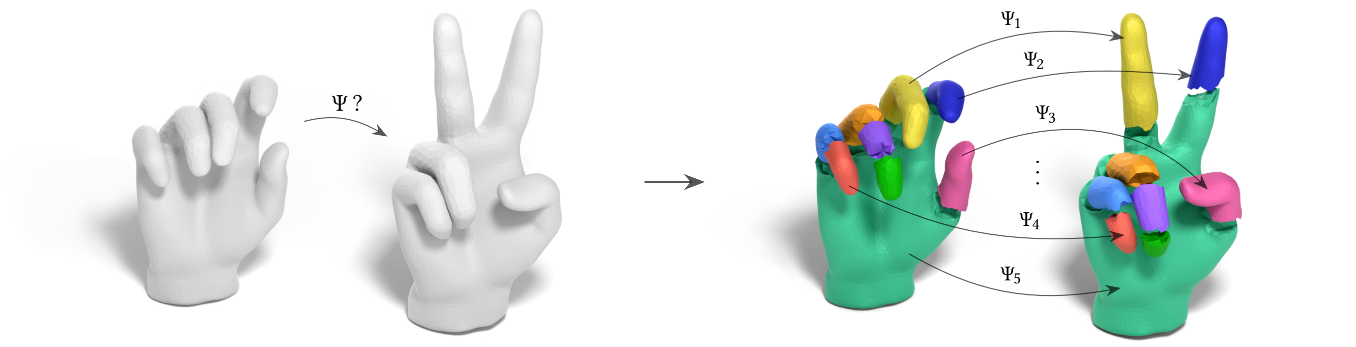

Fig. 1. Our approach addresses the problem of volumetrically mapping a 3D shape onto another shape in a guaranteed *bijective* manner. In contrast to previous work, arbitrary ball-topology shapes are supported. This is achieved by computing a *compatible decomposition* of source and target shape such that the target shape's pieces are *star-shaped*. This enables the piecewise application of recent reliable map computation methods tailored to star-shaped targets.

A method for the construction of bijective volumetric maps between 3D shapes is presented. Arbitrary shapes of ball-topology are supported, overcoming restrictions of previous methods to convex or star-shaped targets. In essence, the mapping problem is decomposed into a set of simpler mapping problems, each of which can be solved with previous methods for discrete star-shaped mapping problems. Addressing the key challenges in this endeavor, algorithms are described to reliably construct structurally compatible partitions of two shapes with constraints regarding star-shapedness and to compute a parsimonious common refinement of two triangulations.

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

Additional Key Words and Phrases: volumetric parametrization, homeomorphism, decomposition, star-shaped

ACM Reference Format:
Steffen Hinderink, Hendrik Brückler, and Marcel Campen. 2024. Bijective Volumetric Mapping via Star Decomposition. *ACM Trans. Graph.* 43, 6, Article 168 (December 2024), 11 pages. https://doi.org/10.1145/3687950

1 INTRODUCTION

Maps between shapes are an important ingredient of numerous methods in computer graphics and geometry processing. Tasks in the fields of parametrization, mesh generation, geometric modeling, shape co-analysis, animation, structure transfer, fitting and comparison hinge upon these. Of particular interest are homeomorphisms, i.e. continuous maps that are *bijective*, as this is required by most use cases.

While the computation of such bijective maps for 2D shapes is well-understood, only recently reliable methods to construct bijective 3D *volumetric* maps have been described. So far, however, they are restricted in terms of their supported class of shapes. Concretely, the target shape is required to be a ball or cube [Campen et al. 2016] or be star-shaped [Hinderink and Campen 2023; Nigolian et al. 2023], i.e. with a non-empty visibility kernel.

We extend this to *any* ball-topology shape, leveraging the above restricted methods in our approach to solve parts of the overall problem. Fig. 1 illustrates the underlying idea that leads to a theoretically sound overall method.

1.1 Method Overview

We assume as input a tetrahedral mesh of the source shape, together with a prescribed boundary homeomorphism onto the surface of the target shape. The desired output is a volumetric homeomorphism into the target shape, conforming with this boundary map, represented in a piecewise linear manner on the source tetrahedral mesh or a refinement thereof.

Our approach to achieve this can be broken down into four logical steps, as illustrated in Fig. 2:

(1) Decompose target shape into star-shaped parts,
(2) Decompose source shape compatibly,
(3) Extend boundary map onto interior part boundaries,
(4) Compute volumetric homeomorphism part by part.

Each step could be instantiated algorithmically in a variety of ways, with different trade-offs between simplicity, efficiency, quality, and robustness. We focus herein on unconditional robustness and simplicity, providing prototypical algorithms for steps (1), (2), (3), and delegating (4) to recent reliable star-shaped mapping algorithms [Hinderink and Campen 2023; Nigolian et al. 2023].

Note that the modular structure provides flexibility for future improvements regarding additional desiderata besides the hard requirement of bijectivity, such as efficiency and map quality, by exchanging algorithms implementing the individual steps.

Authors' address: Steffen Hinderink, sthinderink@uos.de; Hendrik Brückler, hendrik.brueckler@uos.de; Marcel Campen, campen@uos.de, Osnabrück University, Germany.

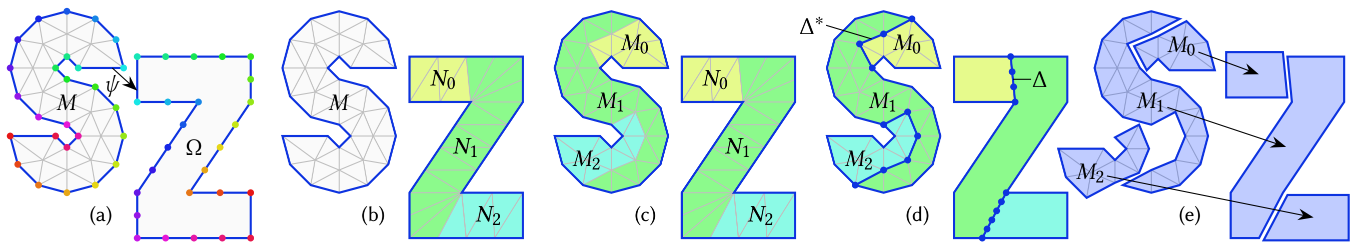

Fig. 2. 2D illustration of our approach. The input (a) is a mesh M with prescribed boundary map, defining target shape Ω. This domain Ω is triangulated/tetrahedrized so as to enable the simple computation of a partition into star-shaped parts $N = \bigcup N_i$ (b). Then a compatible (not necessarily star-shaped) partition $M = \bigcup M_i$ is determined (c) and the boundary map extended onto the interfaces in a piecewise linear manner (d). Finally, the mapping problem can be solved part by part (e), now with a star-shaped target shape in each case.

The main challenges that need our attention lie in steps (2) and (3). In step (2) a decomposition of the source shape needs to be constructed that is combinatorially identical to the target decomposition, while being geometrically embedded in a different shape, the source shape. Along the boundary the decomposition needs to geometrically comply with the prescribed boundary map. In step (3) for each pair of corresponding source part and target part a boundary homeomorphism needs to be constructed, as input for the final step. These part boundary maps need to comply with the global boundary map where applicable and conform across interfaces between parts. A challenge lies in representing these maps. Their piecewise linear representation requires the computation of a common tessellation of source and target part boundaries. We spell out reliable algorithms addressing these challenges for each step in Sections 4 to 7.

Note that, considering the availability of methods for bijectivity-preserving map optimization (in particular towards low distortion) [Rabinovich et al. 2017; Fu et al. 2015], our focus herein lies entirely on establishing *a* valid bijection.

An open source implementation of the method is available at https://github.com/SteffenHinderink/StarDecompositionMaps.

2 RELATED WORK

2.1 Volumetric Mapping

A variety of methods for volumetric map generation based on numerical optimization have been proposed—some targeting low distortion [Aigerman and Lipman 2013; Kovalsky et al. 2014], some harmonicity [Gu et al. 2003], some dedicatedly injectivity [Garanzha et al. 2021; Du et al. 2020; Naitsat et al. 2020; Abulnaga et al. 2023]. While especially some of the latter category have been shown to work extremely well in many scenarios, unfortunately none of these optimization-based methods are able to reliably establish a bijection [Hinderink and Campen 2023, §7.3].

Methods based on combinatorial approaches [Hinderink and Campen 2023; Nigolian et al. 2023; Campen et al. 2016] are able to provide certain guarantees in that regard. Unfortunately, they are all restricted to star-shapes (or even smaller classes of shapes). A recent 2D mapping method [Livesu 2024] paves the way to yet another combinatorial volumetric mapping method—again targeting the star-shape scenario.

The idea of decomposing a complex mapping problem into simpler ones has been employed in various forms, e.g. decomposition into ball-topology pieces [Xu et al. 2013] or into cuboids [Brückler and Campen 2023]. These, however, do not provide a reliable solution for the general mapping problem setting we target here.

2.2 Volumetric Decomposition

The decomposition of geometrically complex domains into parts from a restricted shape class is a classical problem. Convex and star-shaped parts are of particular interest. While the 2D setting has received broad attention, results concerning our 3D setting are relatively scarce.

Known algorithms to compute *minimal* decompositions (of simple polygons) into star-shaped parts are impractical already in the 2D case, with complexity $O(n^{105})$ [Abrahamsen et al. 2024], or $O(n^7 \log n)$ [Keil 2000] when not allowing internal Steiner vertices. This motivates the use of non-minimizing approaches in the 3D setting at hand. Varadhan and Manocha [2005] describe a space partitioning method that could be adopted to our setting, recursively splitting the 3D shape using octree planes until per-part star-shapedness is reached. The resulting parts, however, would not conform with edges of the boundary mesh, instead cutting through triangles, inducing undesirable mesh refinement. Yu and Li [2011] describe a greedy method for mesh based star decomposition. It relies on integer linear programming and a multi-resolution mesh hierarchy, and requires as additional input a shape skeleton whose nodes must guard the entire surface. As (near-)minimality is not of obvious benefit to our approach, we instead use a simpler greedy strategy without preconditions in our method's step (1).

Other works have considered related problems such as *approximate* convex decomposition [Wei et al. 2022; Attene et al. 2008], star-shaped *coverings* (with overlapping parts) [Kawaharada and Sugihara 2003; Yu et al. 2013; Yu and Li 2011], or approximate star-shaped coverings [Lien 2007].

2.3 Volumetric Cutting

In step (2), to form decomposition walls, we need to compute surfaces inside the volume, bounded by prescribed curves. This common problem has been addressed before, defining the surface for instance using implicit splines [Takayama 2019; Haberleitner and Jüttler 2017], linear programs [Grady 2006], via implicit harmonic fields [Gao et al. 2016], or via minimal area or minimal flux principles [Xian et al. 2011]. Unfortunately, while we need surfaces of disk topology, none of these approaches offer control over the topology of the resulting surface. Fig. 6 illustrates this for the case of minimal surfaces, which may not even be topologically unique [Dierkes et al. 2010]. Methods yielding disk topology by construction, e.g. by employing an explicit parametric representation [Haberleitner and Jüttler 2017], respect the boundary curve and the containment requirement only approximately.

Closest to our needs is a topologically safe *patch shift* construction [Brückler and Campen 2023, §6.1.2] that we adapt to our setting in Section 5.2.2.

3 INPUT/OUTPUT

For the sake of precision, let us fix the following formal notation. A *tetrahedral mesh* M is a pure geometric simplicial d-complex with $d = 3$, a *triangle mesh* the same with $d = 2$. Its *carrier* $[M] \subseteq \mathbb{R}^3$ is the union of all its simplices $\bigcup_{s \in M} s \subseteq \mathbb{R}^3$. The *closure* $\langle\cdot\rangle$ of a set of simplices is the smallest simplicial complex containing them. The *boundary* ∂M is the closure of all $d-1$-dimensional simplices in M that are faces of only one d-dimensional simplex. The boundary of a tetrahedral mesh is a triangle mesh and $[\partial M] = \partial[M]$.

Input. The input to our method is a tetrahedral mesh M, such that $\partial[M]$ is 2-manifold and simply connected, i.e. $[M]$ has *ball topology*, together with an orientation-preserving bijective continuous boundary map $\psi : \partial[M] \to \partial\Omega \subseteq \mathbb{R}^3$, linear per boundary triangle. Notice that the target shape Ω is defined via this map. It can be a polyhedron of any shape, it only has to also have ball topology.

Output. The output is a bijective continuous map $\Psi : [M] \to \Omega$ with $\Psi(p) = \psi(p)$ for all $p \in \partial[M]$. It can also be viewed as a *parametrization* of M over Ω. Ψ can be expressed piecewise linearly over a possibly refined version of M.

Numerics. Our method is designed to make use of only simple rational arithmetic operations. In fact, the only important construction involved is the computation of coordinates of centroids of simplices. As such, we support exact rational number types, on the input and on the output side, in line with other recent reliable mapping methods, thereby enabling the avoidance of any numerical risk.

4 STAR DECOMPOSITION

In step (1) we need to decompose Ω into star-shaped parts. Note that a triangulation of its boundary is induced by the image of M's boundary triangle mesh, $\psi(\partial M)$. We proceed as follows to reliably yield such a decomposition in a simple manner.

First, we tetrahedrize Ω, obtaining a tetrahedral mesh N, such that $[N] = \Omega$ and $\partial N = \psi(\partial M)$, i.e. the tetrahedrization conforms to the mapped boundary triangle mesh. Various algorithms are available for the tetrahedrization of polyhedra [George et al. 2003; Si 2015; Hu et al. 2018, 2020; Diazzi et al. 2023]. Note that some of these may refine the boundary mesh ∂N; in such case this refinement needs to be propagated to M to maintain correspondence. In case tetrahedral meshes (with compatible boundary meshes) of source *and* target are already available, this can also be used and the above tetrahedrization be skipped.

We then construct the star decomposition discretely, such that sets of tetrahedra form its parts.

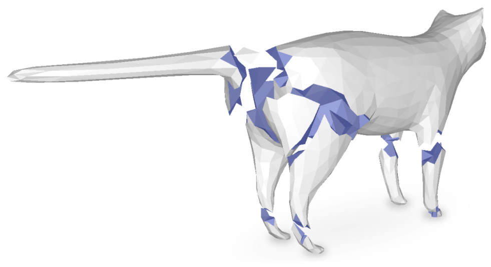
Fig. 3. Exploded-view of a star decomposition of an example model. As all subsequent steps' correctness does not depend on the decomposition's quality, also such very nonuniform decompositions can be used.

4.1 Star-Shaped Submeshes

A set $X \subseteq \mathbb{R}^k$ is *star-shaped* if there exists a *center* $x_0 \in X$, such that $\{\lambda x + (1-\lambda)x_0 \mid \lambda \in (0,1)\} \subseteq X \setminus \partial X$ for all $x \in X$, i.e. the line of sight from the center to any point of x lies entirely inside of X. If X has a piecewise linear boundary, with supporting planes $E_i = \{x \mid {n_i}^\mathsf{T} x = d_i\}$, n_i pointing outwards, this is equivalent to

$$ {n_i}^\mathsf{T} x_0 < d_i \quad \forall i. \tag{1} $$

We wish to decompose the mesh N into submeshes N_i that are *stars*. Formally, we partition the set $C_N = \dot{\bigcup}_i C_i$ of tetrahedra of N into subsets C_i, such that $[N_i]$ with $N_i = \langle C_i \rangle$ is star-shaped.

A naive solution is the maximum partition, with each tetrahedron of N forming a separate (trivially star-shaped) part. To reduce effort and mesh refinement in the subsequent steps, a coarser partition is of benefit, though. We therefore agglomerate adjacent tetrahedra. Starting from an unvisited tetrahedron c, we initialize a star $C_i = \{c\}$. Then, adjacent tetrahedra are iteratively added to C_i, as long as it stays star-shaped. A tetrahedron can be added if it shares at least one triangle with $\langle C_i \rangle$ and there is a center x_0 with respect to which the new star boundary still satisfies equation (1), tested using a linear program [de Berg et al. 1997]. If no more tetrahedra can be added, we start growing the next star from an unvisited tetrahedron. This is repeated until all tetrahedra are assigned, see Fig. 3.

5 COMPATIBLE SOURCE DECOMPOSITION

Given the star decomposition of N, we have to construct a *compatible* decomposition of M, with a part M_i for each part N_i. Compatibility means: The cell complexes formed by the two decompositions have to be isomorphic, i.e. identical in terms of their connectivity structure. Otherwise, per-part homeomorphisms $\Psi_i : [M_i] \to [N_i]$ will not piece together continuously to a global homeomorphic map Ψ. Furthermore, geometrically, the boundary map ψ needs to be respected: If a part M_i lies on the boundary ∂M, i.e. $\partial M_i \cap \partial M \neq \emptyset$, then we need $\psi(\partial M_i \cap \partial M) = \partial N_i \cap \partial N$.

Importantly, though, the parts M_i do not have to be star-shaped. This is because, for step (4), methods exist that can map arbitrarily shaped sources M_i onto star-shaped targets N_i.

We generate a compatible decomposition in a hierarchical manner, akin to a binary space partition. We divide N into two components N^+, N^-, both consisting of a subset of the parts $\{N_i\}$, and both again of ball topology. To this end, the split surface Δ needs to be formed by walls of the star decomposition, to have *disk topology*, and to

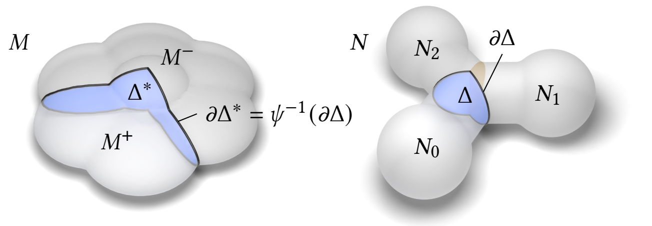

Fig. 4. Illustration of compatible split surfaces in source M and target N.

lie in the interior of N. Only its simple boundary curve $\partial\Delta$ must lie on the boundary ∂N, see Fig. 4. A compatible disk surface Δ^* is then constructed in M. This leaves us with two smaller subproblems (M^+, N^+) and (M^-, N^-). We describe this first split operation in the following, the subproblems can then be handled recursively.

5.1 Split Disk Picking

To identify a split surface, consider a part N_i that lies on the boundary, i.e. $\partial N_i \cap \partial N \neq \emptyset$. Let $\Delta_i = N_i \cap \langle N \setminus N_i \rangle$ denote the triangle mesh that forms the *interface* between N_i and the rest of N. This interface can consist of multiple (edge-connected) components $\Delta_i = \bigcup_j \Delta_{ij}$. We find a part N_i for which at least one component Δ_{ij} has disk topology, and only its boundary curve $\partial\Delta_{ij}$ lies on the boundary ∂N, not any interior point. This surface Δ_{ij} is then used as split surface Δ. See Fig. 4 right for an illustration.

There exist rare star decompositions that do not permit such a part choice. Fig. 5 shows a contrived minimal example. This problem can, however, be addressed by further decomposing the stars into smaller stars. In the extreme, each tetrahedron of N forms its own star; then finding an order in which parts can be picked to define disks is equivalent to finding a simplicial *shelling* order [Dey et al. 1999]. While in theory an exhaustive search and subdivision could be required to that end [Furch 1924], a simple greedy process has proven highly effective [Campen et al. 2016; Hinderink and Campen 2023]. In our experiments, we never even encountered a case where the star decomposition resulting from the algorithm of Section 4 required taking any such measures.

5.2 Finding Compatible Split Disks

In order to split M in a way compatible with how Δ splits N, a corresponding cut surface $\Delta^* \subseteq M$ needs to be defined. For conformance with the boundary map ψ, we need $\partial\Delta^* = \psi^{-1}(\partial\Delta)$, i.e. the

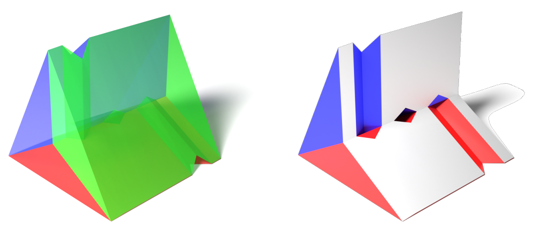

Fig. 5. A polyhedron decomposed into three star-shaped parts. The polyhedron is a triangular prism, with three dead-end tunnels, each bored from an edge up to the opposite star. In this case, each star is interfaced with the rest via an annulus (white in the right interior view) rather than a disk.

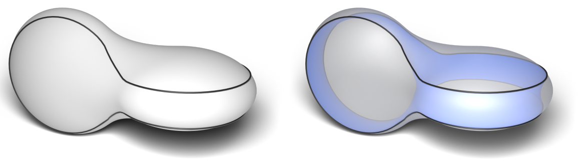

Fig. 6. The blue minimal surface, despite being bounded by a simple curve (black), is not of disk topology, not simply connected. Such a surface cannot be used as split surface in our method.

boundary edge loop of the sought cut surface is known, cf. Fig. 4 left. Furthermore, just like Δ, Δ^* must not touch the boundary of M elsewhere and it must have disk topology.

Note that the cut may require refinement of M. If for example the two incident boundary triangles to an edge of $\partial\Delta^*$ are incident to the same tetrahedron, there is no feasible solution unless this tetrahedron is split. As a preprocessing step, we split every interior edge if both its vertices, and every interior triangle if all its three edges lie on the boundary ∂M.

As discussed in Section 2.3, while yielding nicely shaped results, approaches based on minimal surfaces or harmonic isosurfaces do not offer guarantees regarding disk topology, cf. Fig. 6. We therefore attempt to generate a surface using such an approach (Section 5.2.1), but fall back to a topologically safe approach if disk topology is not achieved (Section 5.2.2). Note that, because M has ball topology, the loop $\partial\Delta^*$ splits the boundary ∂M into two halves, ∂M^+ and ∂M^-, both of disk topology, as illustrated in Fig. 4 left.

5.2.1 *Harmonic Isosurfaces.*

Let L be a Laplacian matrix of the dual of M, in which the tetrahedra are the nodes. A discrete harmonic field x is calculated by solving the Laplace equation $Lx = 0$, with boundary values fixed to $+1$ along ∂M^+ and -1 along ∂M^-. Let those triangles between tetrahedra a and b with $x_a < 0$ and $x_b \geq 0$ form Δ^*. If it is not of disk topology, we fall back to the below alternative.

5.2.2 *Shift Surfaces.*

Notice that the half-boundary ∂M^+ (as well as ∂M^-) is a surface that nearly fulfils our requirements for Δ^*: By definition, it is bounded by the loop $\partial\Delta^*$ and of disk topology. It just does not lie *inside* of M but coincides with its boundary.

Following an idea of Brückler and Campen [2023, §6.1.2] we initialize $\Delta^* = \partial M^+$ and then incrementally shift it into the interior of M, keeping its boundary fixed and maintaining its topology. While the original approach is based on an operator that shifts across a single tetrahedron, we atomically group shifts across multiple adjacent tetrahedra. This allows providing a proof of termination more easily, based on a strictly monotonically decreasing number of remaining contacts between surface and boundary.

Shift Operator. Let the *dome* of a simplex s in the interior of Δ^* be the set of those tetrahedra that lie in front of Δ^* and of which s is a face, as illustrated in Fig. 7. We assume the orientation of Δ^* to be such that its side that initially faces inwards is its *front* side. Our atomic shift operator, for a given simplex s, simply shifts the surface across its dome, thereby moving the surface off of this simplex. Concretely, for the dome of simplex s, define as its *floor*

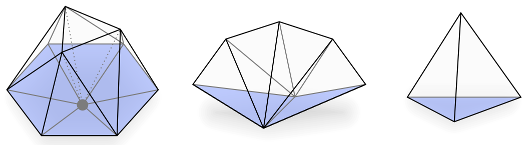
Fig. 7. The domes of a vertex, an edge, and a triangle, respectively. Their floors are marked in blue.

the simplices on its boundary of which s is a face. The complement of the floor's closure in the dome's boundary is the dome's *ceiling*. Note that the floor is part of Δ^*. Shifting Δ^* across the dome means removing its floor from Δ^* and adding its ceiling to Δ^*.

On-Demand Refinement. There are two potential issues with this. First, the ceiling may contain elements from Δ^*. Shifting would then alter the topology of Δ^*. Second, the ceiling may contain elements of the boundary ∂M. Shifting would then not only move off of the boundary, but also onto it again, violating our goal of monotonic progress. Both issues can be circumvented through mesh refinement, selectively inserting an intermediate level between floor and ceiling where necessary.

To this end, we define a simplex s' in the ceiling to be *forbidden* if $s' \in \Delta^* \cup \partial M$. The following refinements by means of splits in the dome's interior are performed, depending on the type of s:

- Vertex v:
  1. Split edges vv' with v' forbidden
  2. Split triangles ve' with e' forbidden
  3. Split tetrahedra vf' with f' forbidden
- Edge e:
  1. Split triangles ev' / ee' with v' / e' forbidden
  2. Split tetrahedra ee' / ef' with e' / f' forbidden
- Triangle f:
  1. Split tetrahedra fv' / fe' / ff' with v' / e' / f' forbidden

After these splits, the shift can safely be performed without reaching the former ceiling in the forbidden spots. Fig. 8 illustrates this.

Greedy Schedule. We apply the dome-based shift operator in a greedy manner, to interior simplices of Δ^* that lie in ∂M, prioritizing vertex-based shifts over edge-based shifts over triangle-based shifts. Note that a vertex-based shift moves Δ^* not only off of the vertex but also off of all incident floor edges and triangles. Therefore the edge-based shift has to come into play in only a few cases (when both incident vertices are in $\partial\Delta^*$). The triangle-based shift comes

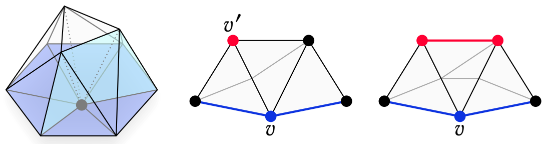

Fig. 8. Splits in the dome of vertex v. The floor is blue, forbidden elements are red. For clarity, splits are shown only on the cyan marked cross section (left). The edge vv' is split (center). If further elements are forbidden, splits are performed successively (right). Note that the two edge splits already remove the forbidden edge from the ceiling, i.e. a triangle split (rule 2) does not come into play here.

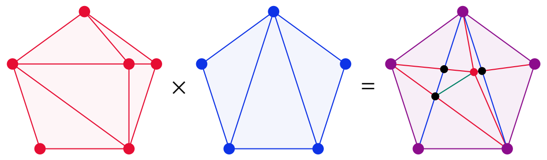

Fig. 9. Two triangle meshes (red and blue) and a common triangulation obtained by overlaying their Tutte embeddings. Elements corresponding to elements of the input are colored respectively. Shared elements are colored in purple. At the intersections of red and blue edges new vertices (black) are created. One quadrilateral is split into two triangles by the green edge.

into play only in the extreme case that $\partial\Delta^*$ consists of only 3 edges. Within each class, we further prioritize shifts based on the number of splits they would cause.

Termination of this algorithm is shown in appendix A.1.

Split Surface Shape Improvement. Once the split surface has been moved off of the boundary completely, we apply further shifts (not requiring refinement) across individual tetrahedra as long as this decreases the surface's discrete area.

6 BOUNDARY MAP EXTENSION

Now that the corresponding surfaces Δ and Δ^* are determined, we extend the boundary map ψ onto them. The requirement is that along the boundary $\partial\Delta$ the boundary map is respected, and that the map is a homeomorphism. As our overall setting requires a piecewise linear representation of this map, the main challenge is due to the incompatible triangulations of these two surfaces. In essence, we need to refine the meshes to a common triangulation. This way, there are pairs of corresponding vertices $v_M \in \Delta$ and $v_N \in \Delta^*$, and we can set $\psi(v_M) = v_N$ to define the map piecewise linearly.

Abstractly speaking, we need to find a triangulation T (of a polygon) that contains two other triangulations. This means that for every element of either triangulation there exists a subset of elements in T such that in total these subsets have the same connectivity in T as the original elements in their respective triangulation.

6.1 Overlay

A simple way to obtain a common triangulation is to bijectively map the two triangle meshes onto the plane with the same boundary. The planar triangle meshes can then be overlaid. The result is a *polygon mesh*, polygons with > 3 vertices are triangulated to obtain T. For the maps a Tutte embedding [Tutte 1963] can be used. Fig. 9 shows an example.

We observe that this straightforward approach, while theoretically sound, comes with two practical downsides. First, the embeddings sometimes degenerate due to numerics. While theoretically the embedding equation system could be solved in exact rational arithmetic, this comes with high computational costs [Shen et al. 2019]. An alternative more efficient numerically safe approach based on Schnyder embeddings [Finnendahl et al. 2023] does not support the arbitrary boundaries that we need to deal with. Second, the overlay often leads to an unnecessarily fine common triangulation.

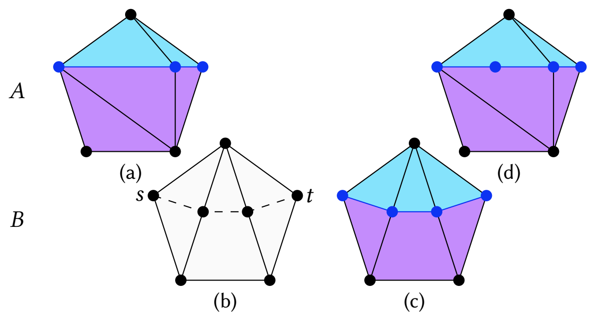

Fig. 10. First step of the recursive alignment algorithm on the same input as in Fig. 9. Mesh A is divided into two halves (a) by a path (blue). Mesh B is refined (b) to enable the path (c, blue) between s and t. One additional vertex is inserted into A, such that both paths consist of four vertices (d).

For this reason we propose another approach in the following, that easily supports exact arithmetic and aims for parsimony.

6.2 Recursive Triangulation Alignment

Similarly to how our overall mapping approach divides one tetrahedral mesh with a surface that is then compatibly formed in the other one, we divide one triangle mesh with a path and determine a corresponding path in the other one. Note that in the case of the volume division, however, we choose a potentially imbalanced split, possibly splitting off a single part (Section 5.1), so as to ensure a simple cut surface. Here, by contrast, we can choose a balanced split for efficiency, because topologically simple split *paths* are easy to construct. By choosing these paths to be aligned with existing edge paths whenever possible, the amount of refinement induced by overlaid misaligned triangles (cf. Fig. 9) is reduced.

For the resulting halves this is recursively repeated in a divide-and-conquer manner until only trivial meshes remain. The recursion step is illustrated in Fig. 10 and explained in detail in the following. Input to the recursion step are two *polygon* meshes with matching boundary that have disk topology; initially, these are Δ and Δ^*.

Source path. Let A be the polygon mesh with more (or the same amount) polygons and B the other one. If the number of polygons in A (and in B) is one, the recursion anchor is reached, they do not need to be divided any further. Otherwise, a path is defined in A that divides it into two parts that both have disk topology.

To obtain a balanced split, an algorithm similar to the one described in Section 5.2.1 can be used, taking an isocurve of a discrete harmonic field. This time, L is a Laplacian matrix of the dual of A, and the field x has a value per polygon. As we have no fixed end points of the sought path to adhere to, a boundary condition free, maximally smooth harmonic field can be computed as the Fiedler vector, i.e. the eigenvector of L corresponding to the second-smallest eigenvalue [Wu et al. 2014]. Instead of the 0-isocurve, we use the median $\tilde{x}$ of the field values as threshold to yield a balanced division. In case $\tilde{x} = \min_i x_i$ or $\tilde{x} = \max_i x_i$, the next larger (smaller) value is chosen as threshold to prevent degeneracy.

The maximum principle implies that the resulting path is simple. But instead of running boundary to boundary, dividing A into two disks, it may sometimes be a cycle, dividing A into a disk and an annulus. In this case we fall back to solving for a discrete harmonic field with boundary condition, fixing two arbitrary boundary values to -1 and $+1$.

Target path. A corresponding path is then sought in B. Let s be the start and t be the end vertex in B, mapped from the path in A via ψ or ψ^{-1}. A path between s and t can either again be computed as an isocurve in a harmonic field or simply as the shortest interior edge path between s and t. This path divides B into the two parts corresponding to those of A. Note that interior refinement of B may be required to enable such a path because the path must not intersect the boundary ∂B except on s and t. This can be achieved as follows:

The vertices s and t divide ∂B into two sides. A *bridge edge* is an interior edge in B that connects one of these sides with the other. We split each bridge edge in two, inserting a new midpoint vertex. Note that this turns initial triangles into general polygons. Further, there may be *bridge polygons*, touching both sides. These block interior paths from s from t. Therefore, each such polygon is split, by inserting an edge (or a sequence of edges), connecting two interior vertices (or s or t) on the polygon's boundary such that the two sides get separated (cf. Fig. 20). Due to the prior splitting of bridge edges such vertices always exist. These splits enable a path from s to t to cross all former bridge edges and bridge polygons.

Path subdivision. Finally, if the numbers of vertices of the two paths differ, the path with the smaller number of vertices is subdivided by inserting additional vertices splitting its edges. This way, the matching boundaries for subsequent recursion steps are established. We distribute these splits as evenly as possible across the path edges.

Final triangulation. In the end, when the recursion is finished, the polygons in both meshes are triangulated equally. Fig. 11 shows an example of the full algorithm.

The termination of the algorithm as well as the importance of using polygon meshes in its intermediate stages, postponing the splitting into triangles until the end, is discussed in appendix A.2.

6.3 Maintaining Conformance

The triangle mesh Δ^* is a submesh of the tetrahedral mesh M. Above we have computed a refinement T of Δ^*, such that ψ could be extended onto it. To make ψ representable on the tetrahedral mesh M as well, it must be refined, too, to make it conform to T. To that end, the two layers of tetrahedra adjacent to the split surface Δ^* (one on its front side, one on the back) are refined analogously, as described in detail in previous work by Hinderink and Campen [2023, §6.2.1]. In brief, tetrahedra adjacent to refined triangles are refined into bouquets of tetrahedra, and tetrahedra adjacent only to refined edges into fans of tetrahedra.

7 PER-PART MAPPING

After the algorithms of Sections 5 and 6 have been applied until termination of the recursion, a bijection per part can be constructed. We have the part pair (M_i, N_i), the boundary map $\psi : \partial[M_i] \rightarrow \partial[N_i]$, and $[N_i]$ is star-shaped. This can be used as input for existing volumetric mapping methods [Hinderink and Campen 2023; Nigolian

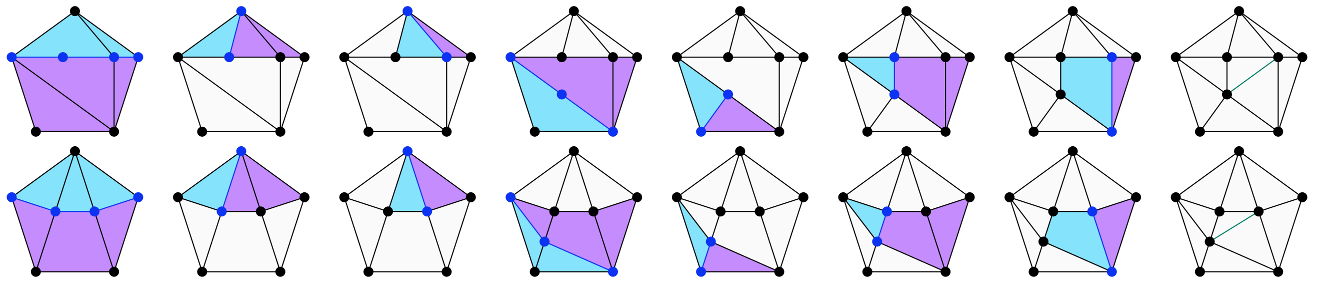

Fig. 11. Recursive alignment of two triangle meshes in the top and bottom row. Input are the same meshes as in Fig. 9, the first step (first column) is also shown in Fig. 10. Paths are marked in blue and the recursively aligned halves in cyan and purple. In the last step (last column) one quadrilateral in both polygon meshes is split into two triangles respectively by the green edges. Note that the resulting common triangulation is smaller than the one in Fig. 9.

et al. 2023] that require the target to be star-shaped and obtain a bijective map $\Psi_i : [M_i] \to [N_i]$.

Finally, the per-part volumetric maps Ψ_i, due to being bijective, internally continuous, and continuous across parts interfaces (due to identical boundary constraints on both interface sides), combine to a continuous bijective global map $\Psi : [M] \to \Omega$.

8 EVALUATION

To test our implementation of the described algorithms, we make use of 600 mapping problem instances, generated by mapping 25 meshes from the MPI MANO data set [Romero et al. 2017] with self-intersections removed (shown in Fig. 12) onto each other (excluding the identities) using the provided boundary correspondence.

Fig. 12. Hand models from the MPI MANO data set. These come with bijective boundary maps.

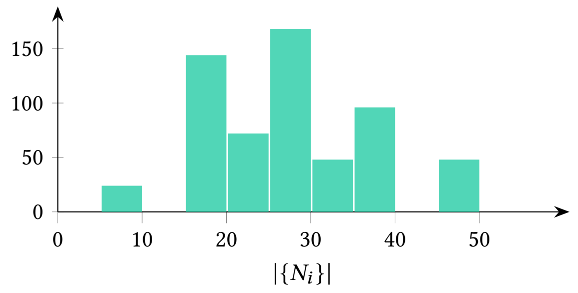

Fig. 13. Histogram of the number of parts in the computed star decompositions.

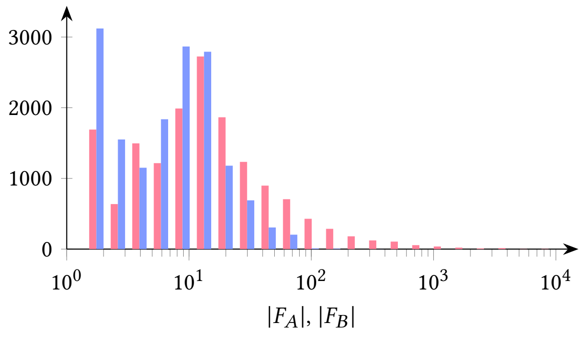

Fig. 14. Histogram of the number $|F_A|$ (red) and $|F_B|$ (blue) of triangles in the larger and in the smaller of the two meshes entering the common triangulation process, over the 16 248 cases.

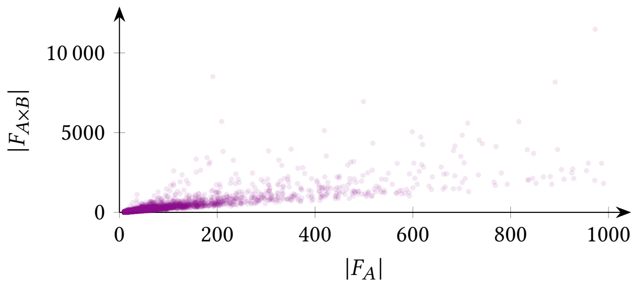

Fig. 15. Each point represents a call of the common triangulation process, indicating the size $|F_A|$ of the larger of the two input meshes and the size $|F_{A\times B}|$ of the output triangle mesh.

Star Decomposition. The star decomposition of the target resulted in 10 to 47 parts per instance, see Fig. 13.

Compatible Decomposition. The split surfaces Δ^* forming compatible source decompositions were computed using the quality-focused harmonic field approach (Section 5.2.1) in 96 % of the cases, in 4 % the method resorted to the reliable shift approach (Section 5.2.2). The total number of split surfaces computed is 16 248.

Common Triangulation. The common triangulation of interface mesh pairs (Δ, Δ^*) is performed using the recursive triangulation alignment process of Section 6.2. Fig. 14 shows the sizes of triangle

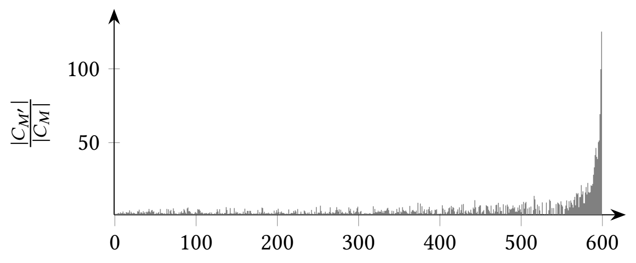

Fig. 16. Overall mesh refinement ratios $|C_{M'}|/|C_M|$, where M' is the output tetrahedral mesh, of the 600 test instances, sorted in the order of Fig. 17.

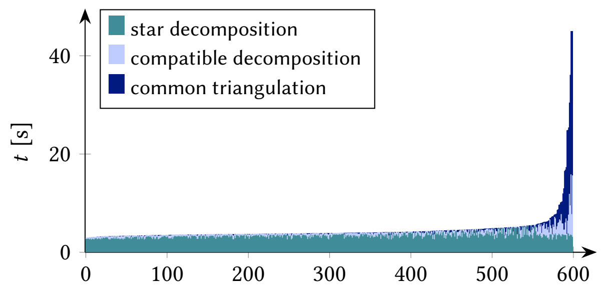

Fig. 17. Run times of our problem decomposition for the 600 test instances, sorted in increasing order. The last three bars reach up to 87 s, 90 s and 150 s. Times were measured on a 2.25 GHz CPU, single-threaded.

meshes entering this process, ranging from 1 to around 14 000 triangles. Regarding the size of the resulting common triangulation, in Fig. 15 we can observe that there is some concentration around the factor of 2 (relative to the larger of the two input triangle meshes), with outliers up to around 45. For comparison, we additionally ran the experiment with the Tutte overlay approach mentioned in Section 6.1. Besides failing in 4 cases due to numerical degeneration, it led to 3.5 % more complex triangulations on average, and took 95× as long.

Refinement. The input meshes M consist of 2600 ± 200 tetrahedra. Our method performs on-demand mesh refinement in some of its steps (in particular those described in Section 5.2.2 and Section 6.2). The refinement ratio (Fig. 16) ranges from 1.1× up to outliers at around 125×, the latter being a clear case of overrefinement (cf. Section 8.1).

Run Time. Fig. 17 reports the times taken per instance to decompose the mapping problem (avg: 5 s), reported separately for the star decomposition computation, the compatible source decomposition, and the boundary map extension. As can be observed, for the slowest instances the largest share is taken by the common triangulation.

Reliability. As mentioned in Section 2.1 and demonstrated before [Hinderink and Campen 2023; Nigolian et al. 2023], numerical mapping methods (in contrast to the here considered constructive setting) have some great features but can lack reliability. The implementations published by Du et al. [2020] and by Garanzha et al. [2021] are unable to yield a fully bijective map for 142 and 53 of

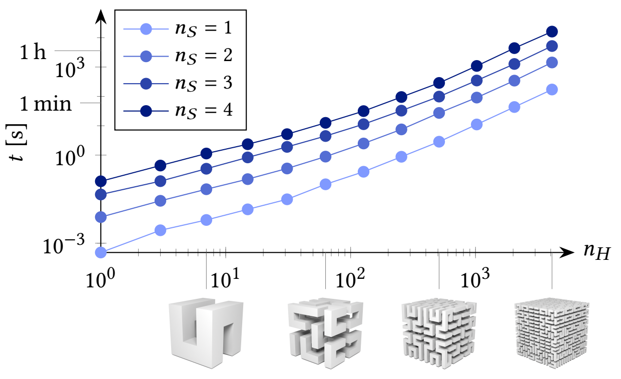

Fig. 18. Run times of our algorithm on problem instances of gradually increasing complexity: A cuboid is mapped to a solid 3-dimensional Hilbert curve. Complexity is controlled by the number n_H of curve segments (i.e. $n_H = 2^{3k} - 1$ corresponds to the order-k Hilbert curve) as well as by the resolution of the boundary mesh, obtained from a maximally coarse mesh by n_S levels of subdivision.

the 600 cases, respectively, even though we assisted by refining the meshes by splitting boundary-to-boundary edges and triangles (otherwise not a single instance was mapped successfully).

Scalability. Our algorithm's run time depends on the resolution of the employed meshes as well as on the geometric complexity of the target shape, which affects the number of star-shaped parts required and the intricacy of the split surfaces. While this dependency is intricate, Fig. 18 attempts to provide at least a sense of it. It shows the run time of our problem decomposition approach for problem instances that were systematically generated to exhibit gradually increasing complexity in these regards.

Further Examples. Further example result maps for a diverse set of instances, including contrived far-from-isometric cases, are shown in Fig. 19 with accompanying information in Table 1.

8.1 Limitations and Future Work

Star Decomposition. Our simple star decomposition (Section 4) is not close to minimal or balanced, cf. Fig. 3. Non-greedy approaches may yield better decompositions. It is, however, not clear whether coarser is better. While few large stars may lead to less effort and refinement in the subsequent steps, small stars lead to simpler per-part mapping problems. Finding a decomposition quality measure (and a tailored decomposition algorithm) would be interesting.

Refinement. Also in the subsequent steps our approach has multiple obvious degrees of freedom. While we here often opted for strategies with a focus on simplicity and correctness for proof-of-concept purposes, a very worthwhile future direction is tuning these degrees of freedom towards minimal mesh refinement. Some amount of refinement can be fundamentally inevitable depending on the target shape geometry, as also discussed by Cherchi and Livesu [2023]. A large share of the refinement performed by our algorithms, however, is likely superfluous, and refinement by orders of magnitude that we sometimes observe certainly is avoidable. Note

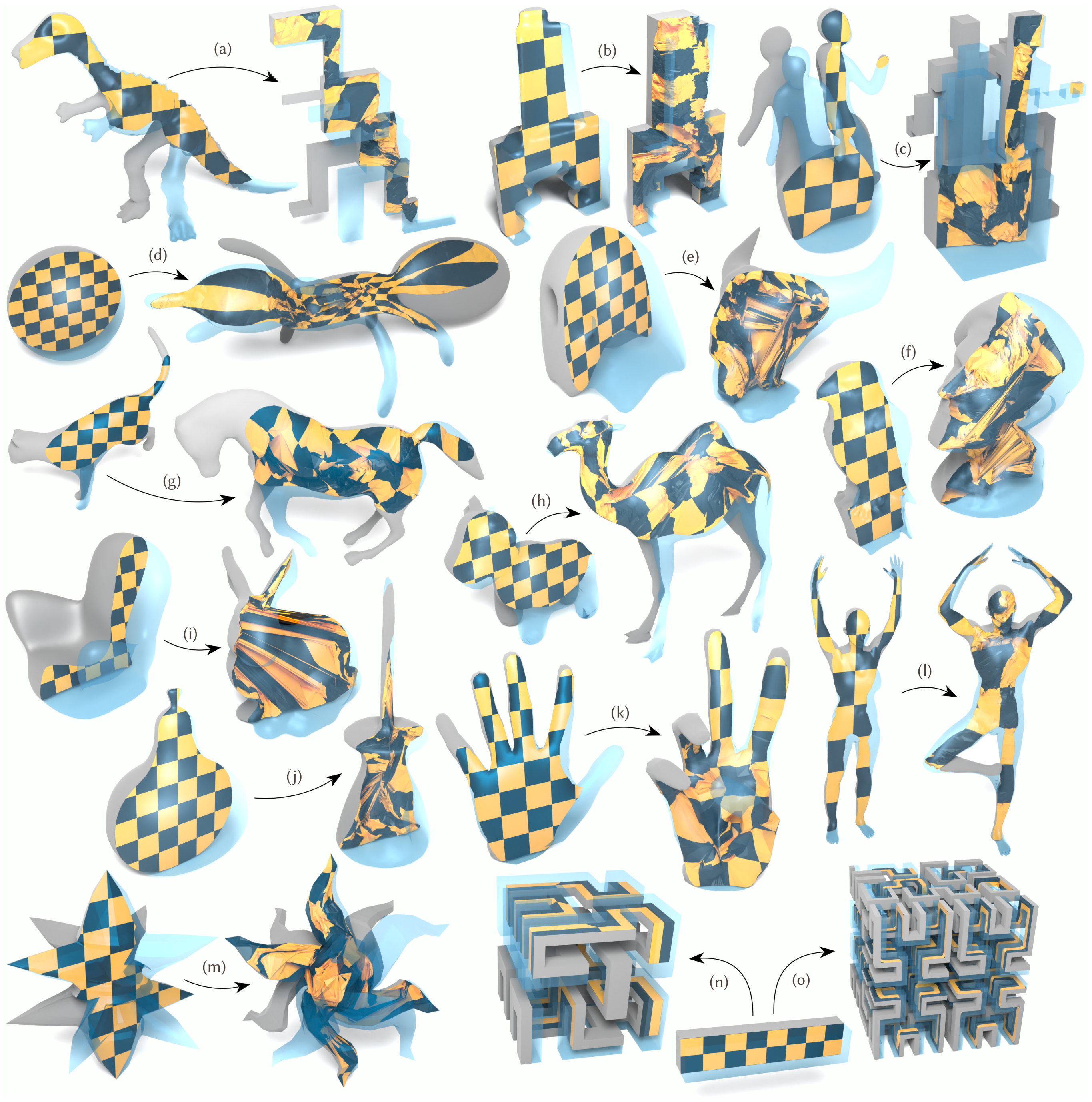

Fig. 19. Visualization of maps constructed by our algorithm on various problem instances, with per-part maps computed using the method of Hinderink and Campen [2023]. The behavior in the interior is depicted by showing the image of a uniform checkerboard pattern on a cross section through each shape. Notice that the maps, while bijective by construction, are often of high distortion. Sources and statistics for each instance are listed in Table 1.

Table 1. Sources, number of parts, refinement ratios, and run times of our problem decomposition approach for the mapping instances depicted in Fig. 19. Boundary maps for instances (e), (f), (i) and (j) were generated using the surface mapping algorithm of Schmidt et al. [2023].

	Source	$\lvert\{N_i\}\rvert$	$\frac{\lvert C_{M'}\rvert}{\lvert C_M\rvert}$	t [s]
(a)	[Du et al. 2020]	40	2.04	135
(b)	[Du et al. 2020]	166	5.37	5292
(c)	[Du et al. 2020]	73	1.46	2355
(d)	[Du et al. 2020]	40	1.61	2068
(e)	[Zhou and Jacobson 2016]	44	2.76	150
(f)	[Zhou and Jacobson 2016]	80	21.3	391
(g)	[Schmidt et al. 2023]	60	1.88	34
(h)	[Schmidt et al. 2023]	148	14.2	130
(i)	[Myles et al. 2014]	83	23.9	506
(j)	[Myles et al. 2014]	23	5.13	49.2
(k)	[Romero et al. 2017]	30	2.98	3.88
(l)	[Bogo et al. 2014]	166	2.77	304
(m)	Self-constructed	154	22.2	246
(n)	Self-constructed	48	1.08	0.098
(o)	Self-constructed	380	1.11	3.03

that this aspect of over-refinement is characteristic also of other recent (less general) constructive mapping methods [Hinderink and Campen 2023; Nigolian et al. 2023; Campen et al. 2016].

Run Time. Note that over-refinement is not just detrimental in terms of parsimony of the output map representation, it also is the main cause for high run times observed depending on the shapes' characteristics. This is due to refinement in early steps increasing the problem sizes in subsequent steps, even causing cascading effects.

Distortion. Our cutting procedure (Section 5) ensures topological compatibility. It does not aim for geometrical similarity. In this sense, the distortion of the constructed map is not in the algorithm's focus (cf. Fig. 19). Bijectivity-preserving map distortion minimization in a post-process [Rabinovich et al. 2017; Fu et al. 2015] is an option, but can be challenging numerically in cases of extreme initial distortion.

Topology. The described method is limited to ball-topology shapes. Note that this hinges solely on the sub-step of finding compatible split surfaces (Section 5.2), for which there is, so far, no known algorithm to reliably generate surfaces of disk topology—except for the described shift approach, which requires ball topology to obtain a disk topology initialization. Extending this to arbitrary topology would be very valuable.

9 CONCLUSION

We have presented a first reliable constructive approach for the computation of volumetric homeomorphisms of ball-topology tetrahedral meshes onto arbitrary target shapes. Partitioning source and target in a compatible way, while restricting to star-shaped target parts, we decompose this hard problem to multiple problems of a simpler class. In this way, we can leverage previous reliable methods. The focus herein being on reliability and simplicity, efficiency-focused variations of the method's three main building blocks (star decomposition, compatible cutting, common refinement) are worthwhile directions for future improvement.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) - 497335132; 456666331.

REFERENCES

Mikkel Abrahamsen, Joakim Blikstad, André Nusser, and Hanwen Zhang. 2024. Minimum Star Partitions of Simple Polygons in Polynomial Time. arXiv:2311.10631

S. Mazdak Abulnaga, Oded Stein, Polina Golland, and Justin Solomon. 2023. Symmetric Volume Maps: Order-invariant Volumetric Mesh Correspondence with Free Boundary. *ACM Trans. Graph.* 42, 3 (2023), 25:1–25:20.

Noam Aigerman and Yaron Lipman. 2013. Injective and Bounded Distortion Mappings in 3D. *ACM Trans. Graph.* 32, 4 (2013), 106:1–106:13.

Marco Attene, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno. 2008. Hierarchical Convex Approximation of 3D Shapes for Fast Region Selection. *Computer Graphics Forum* 27, 5 (2008), 1323–1332.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* 3794–3801.

Hendrik Brückler and Marcel Campen. 2023. Collapsing Embedded Cell Complexes for Safer Hexahedral Meshing. *ACM Trans. Graph.* 42, 6 (2023), 180:1–180:24.

Marcel Campen, Cláudio T. Silva, and Denis Zorin. 2016. Bijective Maps from Simplicial Foliations. *ACM Trans. Graph.* 35, 4 (2016), 74:1–74:15.

G. Cherchi and M. Livesu. 2023. VOLMAP: a Large Scale Benchmark for Volume Mappings to Simple Base Domains. *Computer Graphics Forum* 42, 5 (2023), e14915.

Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. 1997. *Computational Geometry.* Springer.

Tamal K. Dey, Herbert Edelsbrunner, and Sumanta Guha. 1999. Computational topology. In *Advances in Discrete and Computational Geometry.* American Mathematical Society, 109–143.

Lorenzo Diazzi, Daniele Panozzo, Amir Vaxman, and Marco Attene. 2023. Constrained Delaunay Tetrahedrization: A Robust and Practical Approach. *ACM Trans. Graph.* 42, 6 (2023), 181:1–181:15.

Ulrich Dierkes, Stefan Hildebrandt, and Friedrich Sauvigny. 2010. Minimal Surfaces. In *Minimal Surfaces.* Springer, 53–90.

Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z. Kovalsky, Yajie Yan, Danny M. Kaufman, and Tao Ju. 2020. Lifting Simplices to Find Injectivity. *ACM Trans. Graph.* 39, 4 (2020), 120:1–120:17.

Ugo Finnendahl, Dimitrios Bogiokas, Pablo Robles Cervantes, and Marc Alexa. 2023. Efficient Embeddings in Exact Arithmetic. *ACM Trans. Graph.* 42, 4 (2023), 71:1–71:17.

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing Locally Injective Mappings by Advanced MIPS. *ACM Trans. Graph.* 34, 4 (2015), 71:1–71:12.

Robert Furch. 1924. Zur Grundlegung der kombinatorischen Topologie. *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg* 3, 1 (1924), 69–88.

Xifeng Gao, Tobias Martin, Sai Deng, Elaine Cohen, Zhigang Deng, and Guoning Chen. 2016. Structured Volume Decomposition via Generalized Sweeping. *IEEE Transactions on Visualization and Computer Graphics* 22, 7 (2016), 1899–1911.

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. 2021. Foldover-free maps in 50 lines of code. *ACM Trans. Graph.* 40, 4 (2021), 102:1–102:16.

P. L. George, H. Borouchaki, and E. Saltel. 2003. 'Ultimate' robustness in meshing an arbitrary polyhedron. *Internat. J. Numer. Methods Engrg.* 58, 7 (2003), 1061–1089.

Leo Grady. 2006. Computing Exact Discrete Minimal Surfaces: Extending and Solving the Shortest Path Problem in 3D with Application to Segmentation. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 69–78.

Xianfeng Gu, Yalin Wang, and Shing-Tung Yau. 2003. Volumetric Harmonic Map. *Communications in Information and Systems* 3, 3 (2003), 191–202.

Michael Haberleitner and Bert Jüttler. 2017. Isogeometric segmentation: Construction of cutting surfaces. *Computer-Aided Design* 90 (2017), 135–145.

Steffen Hinderink and Marcel Campen. 2023. Galaxy Maps: Localized Foliations for Bijective Volumetric Mapping. *ACM Trans. Graph.* 42, 4 (2023), 129:1–129:16.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4 (2020), 117:1–117:18.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4 (2018), 60:1–60:14.

Hiroshi Kawaharada and Kokichi Sugihara. 2003. Compression of Arbitrary Mesh Data Using Subdivision Surfaces. In *Mathematics of Surfaces.* Springer, 99–110.

J. Mark Keil. 2000. Polygon Decomposition. In *Handbook of Computational Geometry.* North-Holland, 491–518.

Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2014. Controlling Singular Values with Semidefinite Programming. *ACM Trans. Graph.* 33, 4 (2014), 68:1–68:13.

Jyh-Ming Lien. 2007. Approximate Star-Shaped Decomposition of Point Set Data. In *Eurographics Symposium on Point-Based Graphics.* 73–80.

M. Livesu. 2024. Advancing Front Surface Mapping. *Computer Graphics Forum* 43, 2 (2024), e15026.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4 (2014), 135:1–135:14.

Alexander Naitsat, Yufeng Zhu, and Yehoshua Y. Zeevi. 2020. Adaptive Block Coordinate Descent for Distortion Optimization. *Computer Graphics Forum* 39, 6 (2020), 360–376.

Valentin Z. Nigolian, Marcel Campen, and David Bommes. 2023. Expansion Cones: A Progressive Volumetric Mapping Framework. *ACM Trans. Graph.* 42, 4 (2023), 131:1–131:19.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2 (2017), 16:1–16:16.

Javier Romero, Dimitrios Tzionas, and Michael J. Black. 2017. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Trans. Graph.* 36, 6 (2017), 245:1–245:17.

P. Schmidt, D. Pieper, and L. Kobbelt. 2023. Surface Maps via Adaptive Triangulations. *Computer Graphics Forum* 42, 2 (2023), 103–117.

Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. 2019. Progressive Embedding. *ACM Trans. Graph.* 38, 4 (2019), 32:1–32:13.

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2 (2015), 11:1–11:36.

Kenshi Takayama. 2019. Dual Sheet Meshing: An Interactive Approach to Robust Hexahedralization. *Computer Graphics Forum* 38, 2 (2019), 37–48.

W. T. Tutte. 1963. How to Draw a Graph. *Proc. Lond. Math. Soc.* s3-13, 1 (1963), 743–767.

Gokul Varadhan and Dinesh Manocha. 2005. Star-shaped Roadmaps - A Deterministic Sampling Approach for Complete Motion Planning. In *Proc. Robotics: Science and Systems.*

Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. 2022. Approximate Convex Decomposition for 3D Meshes with Collision-Aware Concavity and Tree Search. *ACM Trans. Graph.* 41, 4 (2022), 42:1–42:18.

Jian-ping Wu, Jun-qiang Song, and Wei-min Zhang. 2014. An efficient and accurate method to compute the Fiedler vector based on Householder deflation and inverse power iteration. *J. Comput. Appl. Math.* 269 (2014), 101–108.

Chuhua Xian, Shuming Gao, and Tianming Zhang. 2011. An approach to automated decomposition of volumetric mesh. *Computers & Graphics* 35, 3 (2011), 461–470.

Huanhuan Xu, Wuyi Yu, Shiyuan Gu, and Xin Li. 2013. Biharmonic Volumetric Mapping Using Fundamental Solutions. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (2013), 787–798.

Wuyi Yu, Maoqing Li, and Xin Li. 2013. Optimizing Pyramid Visibiliy Coverage for Autonomous Robots in 3D Environment. In *Proc. 8th International Conference on Computer Science & Education.* 1023–1028.

W. Yu and X. Li. 2011. Computing 3D Shape Guarding and Star Decomposition. *Computer Graphics Forum* 30, 7 (2011), 2087–2096.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. arXiv:1605.04797

A PROOFS

A.1 Surface Shift

Lemma A.1. *The algorithm described in Section 5.2.2 terminates and yields a surface Δ^*, such that Δ^* has disk topology, $\partial\Delta^*$ is the sought boundary, and $\Delta^* \cap \partial M = \partial\Delta^*$ (it does not touch the boundary of M elsewhere).*

Proof. The initial surface ∂M^+ has disk topology and the correct boundary. These are invariants of the algorithm: Each dome is star-shaped, thus of ball topology. Hence each shift across a dome replaces a disk in Δ^* (the floor) by another disk (the ceiling), thereby preserving the surface's topology. Simplices in $\partial\Delta^*$ are not part of the floor of any dome considered, as no interior simplex is a face of them. As only floors are removed from Δ^*, the boundary $\partial\Delta^*$ remains unaltered.

It remains to show that the surface eventually does not touch the boundary of M except for in $\partial\Delta^*$, i.e. that the cardinality of the set $(\Delta^* \cap \partial M) \setminus \partial\Delta^*$ reaches zero. The atomic shift operator based on simplex s, by construction, removes at least s from this set and does not introduce any other. Hence this cardinality decreases strictly monotonically. As long as it is not zero, the set contains a simplex s from the interior of Δ^*, such that a shift based on this simplex can be applied. □

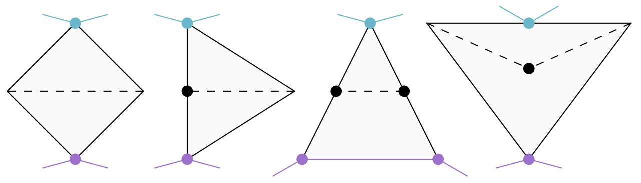

Fig. 20. Splits of bridge polygons into two pieces. The colored elements lie on different sides of ∂B. New vertices (black) are inserted by previous bridge edge splits (center) or as part of an edge sequence (right).

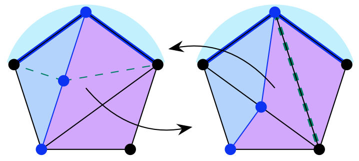

Fig. 21. Infinite loop that could occur when *triangle* meshes were used in the recursive alignment algorithm. The edge split on the left creates the dashed green edges. The resulting purple sub-meshes then have the same connectivity as the input again. For clarity, the path (blue) and split edge (dashed green) of the previous iteration are marked in bold.

A.2 Triangulation Alignment

Lemma A.2. *The algorithm described in Section 6.2 terminates.*

Proof. We prove termination by induction on the number $n = |F_A| + |F_B|$ of polygons. If $n = 2$, the recursion anchor is reached and the algorithm terminates immediately. For the induction step, assume that the algorithm terminates if the number of polygons is less than n. Let A^+ and A^- be the two parts that A is split into in the recursion step, and B^+ and B^- the two corresponding parts of B. In A, only edges are split which does not change the number of polygons, so $|F_{A^+}|+|F_{A^-}| = |F_A|$ and therefore $|F_{A^+}|, |F_{A^-}| < |F_A|$. In B, bridge polygons are split in two and the resulting halves assigned to B^+ and B^-, respectively, so $|F_{B^+}|, |F_{B^-}| \leq |F_B|$. The method is called recursively for A^+ and B^+ as well as for A^- and B^-. Inserting the inequalities per summand yields $|F_{A^+}| + |F_{B^+}|, |F_{A^-}| + |F_{B^-}| < |F_A| + |F_B| = n$, so the algorithm terminates. □

Note that this argument assumes that bridge polygons are split in a way that leads to two pieces only. Fig. 20 illustrates this. Convex polygons can be split in two by inserting a single edge, more general polygons (which occur very rarely in the process) may require a sequence of edges. In our implementation, for simplicity, we instead split such rare polygons into convex pieces.

Further, let us remark that if one were to operate with triangle meshes throughout, always triangulating emerging polygons right away instead of postponing this to after termination of the algorithm, termination can be hampered, cf. Fig. 21.