Galaxy Maps: Localized Foliations for Bijective Volumetric Mapping

STEFFEN HINDERINK and MARCEL CAMPEN, Osnabrück University, Germany

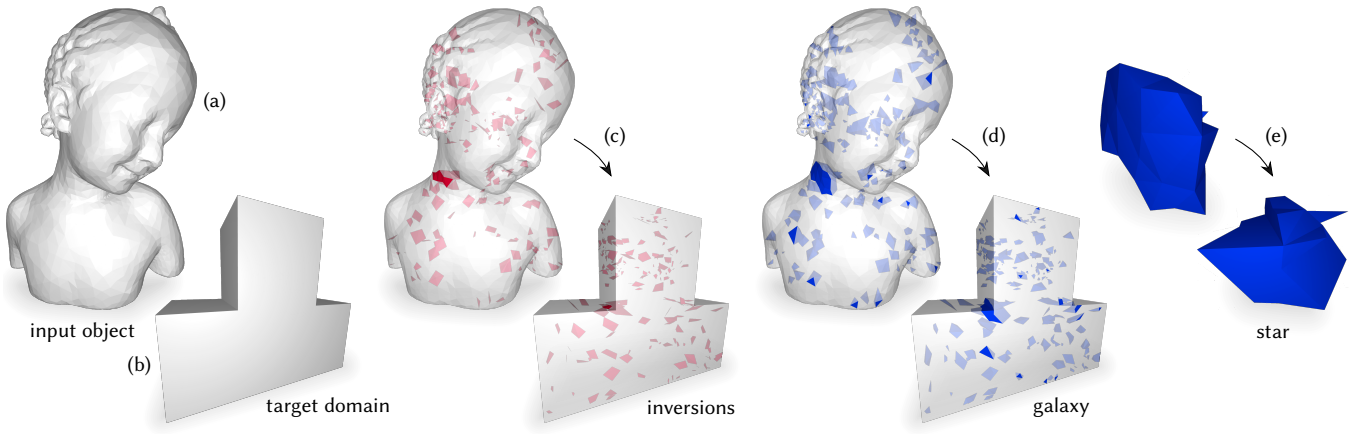


Fig. 1. Our method takes as input a 3D object (a) together with a bijective continuous map of its boundary onto the boundary of an arbitrary convex or star-shaped domain (b). It outputs a bijective continuous volumetric map of the interior, respecting the prescribed boundary map. The core algorithm can be applied either globally, or locally to effectively repair defects (c, red) of non-bijective maps generated by existing non-reliable techniques (here a 3D Tutte embedding). The local approach identifies a set of stars, star-shaped sub-domains, collectively referred to as galaxy (d, blue), that enclose all defects. The algorithm is then efficiently applied star by star (e), so as to locally adjust the map by a bijective replacement while maintaining continuity.

A method is presented to compute volumetric maps and parametrizations of objects over 3D domains. As a key feature, continuity and bijectivity are ensured by construction. Arbitrary objects of ball topology, represented as tetrahedral meshes, are supported. Arbitrary convex as well as star-shaped domains are supported. Full control over the boundary mapping is provided. The method is based on the technique of simplicial foliations, generalized to a broader class of domain shapes and applied adaptively in a novel localized manner. This increases flexibility as well as efficiency over the state of the art, while maintaining reliability in guaranteeing map bijectivity.

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

Additional Key Words and Phrases: volumetric parametrization, homeomorphism, star-shaped, foliation, shelling, tetrahedral mesh

ACM Reference Format:
Steffen Hinderink and Marcel Campen. 2023. Galaxy Maps: Localized Foliations for Bijective Volumetric Mapping. *ACM Trans. Graph.* 42, 4, Article 129 (August 2023), 16 pages. https://doi.org/10.1145/3592410

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

1 INTRODUCTION

Maps between 3D objects as well as volumetric parametrizations over 3D domains are important ingredients in numerous geometric processing and analysis tasks. A common requirement, besides continuity, is that of bijectivity. Application scenarios where this is of relevance include volumetric mesh generation [Brückler et al. 2022a,b; Mandad et al. 2022; Nieser et al. 2011; Pietroni et al. 2022], spline modeling [Martin et al. 2008; Zhang et al. 2012], registration, fitting, and comparison [Iyengar et al. 2012; Martin et al. 2012; Wang et al. 2004], structure or texture transfer [Abulnaga et al. 2023; Li et al. 2007; Lin et al. 2015; Schüller et al. 2013], and interpolation or morphing [Alexa et al. 2000; Xia et al. 2010], to name a few. In these bijectivity (or local or global injectivity) is at least strongly desired, in many cases mandatory to enable reliable processing.

In a 2D setting, convex combination mapping, also called Tutte embedding [Floater 1997; Tutte 1963], is used in countless applications to establish maps of triangle meshes that are bijective by construction, often as initialization subject to further optimization. It is well known that this does not extend to 3D [Floater and Pham-Trong 2006], except for a very restricted class of tetrahedral meshes of, unfortunately, little practical relevance [Alexa 2023].

Discrete conformal mapping is also efficiently possible in 2D due to recent developments [Campen et al. 2021; Gillespie et al. 2021]. Also this avenue does not extend to 3D; in particular, the space of conformal maps is too restrictive to be useful in a generic 3D setting.

Instead, a variety of best-effort approaches are commonly used when it comes to the 3D mapping or parametrization problem (see Section 2). These may or may not succeed in yielding a bijective map on a case by case basis. An exception is a method based on so-called simplicial foliations [Campen et al. 2016], which comes

with guarantees regarding result validity in terms of continuity and bijectivity. It bears strong restrictions, however, regarding supported mapping domain shapes, and has limitations in terms of efficiency in comparison to modern best-effort techniques.

1.1 Contribution

Adopting the central idea of this previous approach [Campen et al. 2016], i.e. reliably constructing bijective volumetric maps using foliations, we make two key contributions generalizing and improving it in particular with respect to flexibility and efficiency:

First, the original work supports only two domains, the sphere and the cube. The latter case furthermore has the downside of not supporting full control over the boundary mapping. Our method supports mapping of arbitrarily shaped ball-topology 3D objects onto arbitrary convex and (even more generally) arbitrary star-shaped domains while offering full support for boundary constraints.

Second, we describe an escalating (multi-tiered) strategy for map construction. Instead of globally, it applies the foliation-based technique in a localized manner, adaptively treating bijectivity issues of other methods used for initialization. This leads to significant benefits in terms of efficiency, even reducing run time by multiple orders of magnitude in some cases.

While already useful on their own, the true benefit actually comes into effect when combining these two techniques and their advantages. Concretely, the support for star-shaped domains enables a particularly efficient form of localization. In turn, the escalating localized approach increases the practical feasibility of treating complex cases and complex domains. Fig. 1 illustrates the method on an example mapping problem.

The method is carefully designed to be implementable in exact rational arithmetic, so as to enable full reliability—notwithstanding that truncation of the resulting map to standard floating point numbers may invalidate guarantees, as discussed further in Section 5.1.1.

We remark that our focus herein is on reliably offering bijectivity, for the concrete setting of ball-topology objects and star-shaped domains, rather than matching or even outperforming alternative non-guaranteed approaches in terms of generality or speed.

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

2 RELATED WORK

We focus our discussion on the discrete 3D volumetric setting and on the aspect of bijectivity. In this realm, pertinent methods can roughly be classified into three categories:

- Methods that optimize maps, *aiming* for bijectivity, either as a byproduct of distortion minimization or dedicatedly.
- Methods that optimize maps, *maintaining* bijectivity, assuming a bijective initialization.
- Methods that constructively *establish* a bijective map in an explicit manner.

2.1 Aiming for Bijectivity

The main cause of non-bijectivity are inversions, i.e. parts of the mesh fold over under the map, their images having negative (or zero) volume. This is often accompanied by high map distortion. Optimizing a map for low distortion can thus lead to inversions vanishing. Depending on the concrete distortion objective employed this can be more or less likely [Aigerman and Lipman 2013; Kovalsky et al. 2014], but there generally is no guarantee. Recently a number of methods have been proposed that, by modifying or augmenting distortion objectives, put a stronger focus particularly on inversion removal or prevention [Abulnaga et al. 2023; Du et al. 2020, 2022; Garanzha et al. 2021; Naitsat et al. 2020; Overby et al. 2021; Poya et al. 2023; Su et al. 2019]—with, in some cases, impressive success rates, but still without a guarantee. As demonstrated in Section 7 this shortcoming is not just of hypothetical nature.

Furthermore, untangling methods [Escobar et al. 2003] from the field of mesh generation are related in that they can be applied to the mesh's image in parameter space under some initial non-injective map, aiming to remove inversions. Noteworthy in this context is the method of Toulorge et al. [2013] which, like our method, uses a localized approach, albeit in a heuristic manner, without a careful selection that would ensure feasibility; it furthermore is based on nonconvex numerical optimization without a success guarantee, whereas we use a reliable constructive method.

2.2 Maintaining Bijectivity

In case a bijective initial map is available, barrier terms can be included in objective functions employed in subsequent processes that, e.g., optimize for lower distortion or improved alignment [Fu et al. 2015; Rabinovich et al. 2017; Schüller et al. 2013; Smith et al. 2019]. These prevent the optimization from inverting any element, by infinitely penalizing any path leading into such a state.

In case the domain shape is subject to optimization as well (*free boundary* setting) global overlaps can additionally be prevented explicitly [Fang et al. 2021; Jiang et al. 2017; Smith and Schaefer 2015; Su et al. 2020], so as to maintain global (not just local) injectivity, and thereby bijectivity onto the resulting domain. All these techniques require a bijective initialization to output a bijective map; one can try to obtain it using one of the above methods, or employ a constructive method that builds a valid starting point, as discussed in the following.

2.3 Establishing Bijectivity

In the 2D setting, an important role is played by methods that, instead of focusing on application-specific map quality in terms of properties like distortion, dedicatedly target bijectivity. Reliable discrete harmonic [Floater 1997; Tutte 1963] and discrete conformal [Campen et al. 2021; Gillespie et al. 2021] techniques are prime examples. Their results are rarely used right away, but considered initializations for subsequent bijectivity maintaining optimization. In the 3D setting, to the best of our knowledge, there is a single method [Campen et al. 2016] (and concurrent work [Nigolian et al. 2023]) that falls into this category; this is in line with a recent survey [Fu et al. 2021] that, regarding the 3D setting, also only mentions this one work. This method is based on the idea of computing a so-called foliation, a partition into a two-parameter family of curves, of the object to be parametrized. This allows reducing the 3D parametrization problem to simpler 2D and trivial 1D parametrization problems, as detailed in Section 4.2. Such foliations (or fibrations) have also been

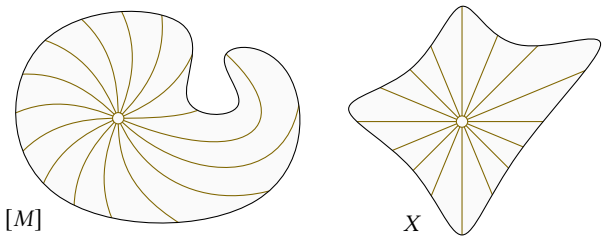


Fig. 2. 2D illustration of a foliation of an (arbitrary, disk-topology) object $[M]$ and a (straight-line) foliation of a star-shaped domain X. Once these foliations (consisting of curves or lines called *leaves*) are constructed, and a boundary map $\psi : \partial[M] \to \partial X$ is prescribed, a bijection $\Psi : [M] \to X$ can be defined in a careful leaf-by-leaf manner.

used in related contexts [Cohen and Ben-Chen 2019; Huynh and Gingold 2015; Jüttler et al. 2019; Trautner et al. 2021]. Our method builds on the underlying idea, generalizing and improving it.

3 OVERVIEW

Before diving into the technical details, let us delineate the proposed method on an intuitive level.

Input: The input to our method is a tetrahedral mesh M, discretizing a 3D object $[M] \subseteq \mathbb{R}^3$ that is topologically a ball, together with a bijective continuous boundary map $\psi : \partial[M] \to \partial X \subseteq \mathbb{R}^3$, i.e. a prescription of the desired image of its surface. We only require the thus prescribed domain X to be star-shaped (which in particular includes any convex domain), as detailed in Section 4.

Output: The result is a bijective continuous map $\Psi : [M] \to X$, also referred to as parametrization of M over the domain X. This map can be output either as an opaque function that can be evaluated for any point $p \in [M]$, or in a piecewise linear manner, i.e. as a set of mapping coordinates ((u, v, w)-coordinates per vertex) on (a refined version of) the mesh M.

Algorithm: We construct a discrete radial foliation of $[M]$, based on a shelling of M, using the technique of Campen et al. [2016]. See Fig. 2 for an intuitive illustration; a precise definition of these concepts follows in Section 4.2. In addition, a point $x_0 \in X$ is computed such that the line segments from x_0 to all points in ∂X form a radial foliation of X; by star-shapedness, it is guaranteed to exist. Via the surface map ψ, the curves and lines forming the two foliations are put into one-to-one correspondence. This correspondence can then be leveraged to, in combination with suitably constructed curve to line maps, define a complete continuous bijection between $[M]$ and X. The details of this algorithm are described in Section 5. It can already be considered a solution to the targeted problem: It produces the desired output exactly as defined above. However, instead of applying it directly, we propose the following more efficient method, which makes use of this algorithm to solve subproblems only.

Method: When best-effort approaches (see Section 2.1), including the simple Tutte embedding approach, fail in computing a map $\Phi : [M] \to X$ that is bijective, they do not yield nothing, but a non-bijective map. Violations of bijectivity often are local in these maps. Our method takes such an imperfect map Φ, virtually cuts out carefully chosen small pieces of $[M]$ and X such that on the remainder the map Φ acts bijectively. In Fig. 1 these pieces are highlighted for an example case. The above algorithm is then applied per piece, subject to boundary constraints that ensure continuity between the resulting piece map and the restricted global map Φ. The union of all these maps then forms the desired bijection Ψ. The description of this method follows in Section 6.

For practical purposes we may even go a step further and establish an escalating (multi-tier) mapping strategy: Start by applying a quick method such as Tutte embedding; if the result is bijective, output this map. Otherwise, escalate to the next tier: Apply a more advanced method (to the initial non-bijective map) that aims for inversion removal, e.g. those discussed in Section 2.1, possibly with a fixed time budget; if the result is bijective, output this map. Finally, if there are still bijectivity issues, apply our (computationally more demanding) method to fix the remaining local issues of the previous tier's result, reliably yielding a bijective map in the end in any case.

4 BACKGROUND

4.1 Meshes

For the purpose of this article, a *tetrahedral mesh* is a pure geometric simplicial complex in 3D, a *triangle mesh* the same in 2D. Let $M = V \cup E \cup F \cup C$ be a tetrahedral mesh with V, E, F, C being the sets of simplices that are its vertices, edges, triangles, and tetrahedra, respectively, each being a compact subset of $\mathbb{R}^3$. The *closure* $\langle\cdot\rangle$ of a subset of simplices is the smallest simplicial complex containing them, i.e. for each contained simplex it also contains all simplices that form its faces; in particular $M = \langle C \rangle$.

For formal precision, we distinguish the mesh M (a collection of simplices) from its *carrier* $[M] = \bigcup_{c \in C} c$ (the subset of $\mathbb{R}^3$ occupied by it). The *surface* ∂M of M is the closure of those triangles in F that are faces of only one tetrahedron. This surface is a triangle mesh, and its carrier $[\partial M] = \partial[M]$.

Our method assumes as input a tetrahedral mesh M such that $[M]$ is 3-manifold with a simply-connected 2-manifold boundary $\partial[M]$ of genus 0, i.e. it has *ball topology*.

4.2 Foliations & Shellings

A *foliation* $\mathcal{F}$ is an (infinite) partition of a manifold into submanifolds [Moerdijk and Mrčun 2003]. We are here concerned in particular with the case of partitioning a 3-manifold into 1-manifolds, i.e. curves. These curves are called *leaves* of the foliation. We will make use of *radial* foliations, in which all leaves start at the boundary (i.e. the surface) and converge to a common point p_0, called the *center* of the foliation, in the interior. Formally, in this case the curves foliate the 3-manifold with the center point removed.

As shown by Campen et al. [2016], in the discrete setting of a tetrahedral mesh, a piecewise linear foliation can conveniently be represented by a piecewise constant direction (i.e. unit vector) field, constant per tetrahedron. The leaves are formed by the direction field's integral curves, as illustrated in Fig. 3. It was shown that there is a sufficient condition of combinatorial kind that allows for the generation of direction fields whose integral curves actually form a partition and thereby imply a foliation. For radial foliations this

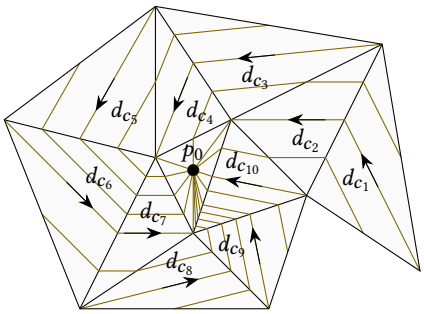


Fig. 3. Illustration of the piecewise linear foliation implied by a piecewise constant vector field in a 2D setting on a triangle mesh.

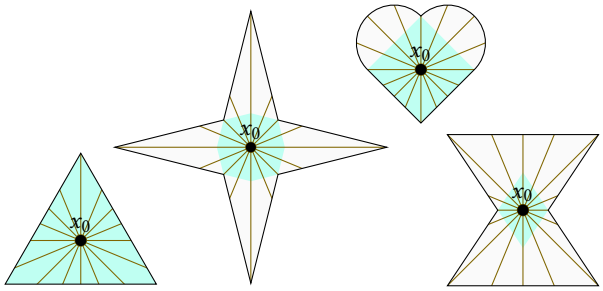


Fig. 4. 2D examples of star-shaped sets and their kernels. Natural foliations for choices of guards are illustrated by a few selected leaves.

boils down to finding a so-called *shelling*, an order $c_1, c_2, \ldots, c_{|C|}$ of the tetrahedra of the tetrahedral mesh. Such orders are guaranteed to exist (possibly after subdivision) and it was demonstrated that in practice a simple greedy algorithm seems sufficient to find one, except for an example contrived by Furch [1924]. Formally, to have a provably complete approach, a fallback to an exhaustive search would need to be added. But also in our experiments on tens of thousands of problem instances we did not encounter a case where the greedy algorithm was insufficient.

Algorithm 1 lists this algorithm, simplified to our radial setting. In this a tetrahedron is to be considered *free* if it has a triangle face in the surface $\partial\langle C\rangle$ and removing it from C leaves $[\langle C\rangle]$ 3-manifold. In the end, the implied piecewise linear foliation is easily extended into the final tetrahedron $c_{|C|}$ that then contains the foliation's center p_0. Note that the original version of this algorithm [Campen et al. 2016, Alg. 1] is more complex as it additionally supports Cartesian foliations, which requires handling of 'parallel' special cases as well as ensuring the manifoldness also of the complement of C throughout—because a bishelling (rather than a shelling) is required. Here we can restrict to the simpler case, while at the same time arriving at a more general method in the following (Section 5).

Algorithm 1: Simplicial Radial Foliation Construction

Input: Tetrahedral mesh $M = V \cup E \cup F \cup C$
Output: Direction field d implying a radial foliation

for $i = 1, \ldots, |C| - 1$
  Select a *free* tetrahedron $c_i \in C$
  Choose direction d_{c_i} such that for each of the triangle faces f of c_i with outwards pointing normal n
    - $\langle d_{c_i}, n\rangle < 0$ if $f \in \partial\langle C\rangle$
    - $\langle d_{c_i}, n\rangle > 0$ otherwise
  $C = C \setminus \{c_i\}$
return d

4.3 Star-Shapes

A set $X \subseteq \mathbb{R}^k$ is *star-shaped* if there exists a point $x_0 \in X$, such that for every $x \in X$ the (open) line section $\{\lambda x + (1-\lambda)x_0 : \lambda \in (0,1)\}$ is contained in $X \setminus \partial X$, the interior of X. In other words, from x_0 every point in X is *visible*, and x_0 is referred to as a *guard* of X. The set of all guards of a set is called the set's *kernel*. Fig. 4 illustrates the concept. Note that there also is a weak notion of star-shapedness (only requiring line section containment in X) but we herein only use the above strict notion. Also note that every non-degenerate *convex* set is star-shaped—in this case the kernel is the entire set.

If X has a piecewise linear boundary—the case relevant for our purposes—note that X is star-shaped with respect to guard x_0 *iff* ${n_i}^\mathsf{T} x_0 < {n_i}^\mathsf{T} a_i$ for all outwards pointing normal vectors n_i of the linear boundary pieces, where a_i is a point on the respective piece.

For a star-shaped set X and a fixed guard point x_0 we define the *natural* radial foliation with center x_0 as the set of all (half-open) visibility line segments $\{\{\lambda x + (1-\lambda)x_0 : \lambda \in (0,1]\} : x \in \partial X\}$.

5 BIJECTIVE STAR-SHAPED MAPPING

We now turn to our algorithm for the construction of a continuous bijection between $[M]$ and a star-shaped domain X.

We follow the idea of Campen et al. [2016] and use a foliation $\mathcal{F}_M$ of $[M]$. Note that their method, however, does not solve our problem: By interpreting per-point values deduced from the foliation as Cartesian or polar coordinates, it maps specifically to Cartesian product domains, in particular $S^2 \times [0,1]$ (*ball map*) and $[0,1]^2 \times [0,1]$ (*cube map*)—and in the latter case only part of the boundary map can be controlled. We instead more flexibly want to enable mapping onto arbitrary convex and star-shaped domains $X \subseteq \mathbb{R}^3$ together with full boundary map prescription capabilities.

To this end, in addition to a radial foliation $\mathcal{F}_M$ of $[M]$, we consider a natural foliation $\mathcal{F}_X$ of X, centered at x_0, as in Fig. 5. Due to the radial nature, the boundary $\partial[M]$ is a *section* of the foliation $\mathcal{F}_M$ (i.e. it has a unique intersection with every leaf) and ∂X is a section of the foliation $\mathcal{F}_X$. Therefore, the given continuous bijective boundary map $\psi : \partial[M] \to \partial X$ establishes a bijection between the leaves of $\mathcal{F}_M$ and the leaves of $\mathcal{F}_X$, which we exploit in the following.

5.1 Map Definition

For a point $p \in [M]$ (except the foliation center p_0) let $l(p)$ be the unique leaf of foliation $\mathcal{F}_M$ that p lies on, and $s(p)$ the unique point where this leaf $l(p)$ intersects the boundary $\partial[M]$, i.e. $s(p) = l(p) \cap \partial[M]$, see Fig. 5. For $p = p_0$ an arbitrary leaf can be picked. The leaf of $\mathcal{F}_X$ corresponding to $l(p)$ is the one that intersects the point $\psi(s(p))$. It is the line segment between x_0 and $\psi(s(p))$. Let $t(p) \in [0,1]$ be the relative position (ratio of lengths) of p along

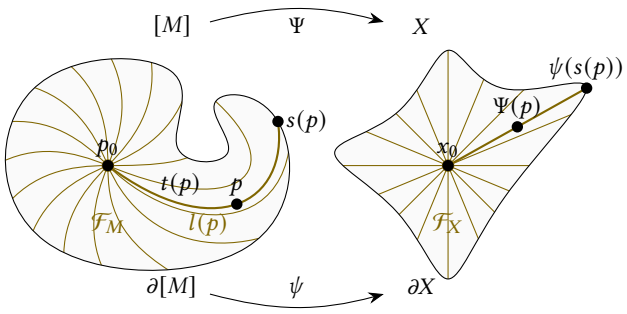


Fig. 5. Defining a bijection (see Section 5.1) between two objects based on compatible foliations $\mathcal{F}_M$, $\mathcal{F}_X$ and a bijective boundary map ψ.

$l(p)$, from 0 at the center p_0 to 1 at the boundary $\partial[M]$. With this, given the surface map ψ, we define our volume map Ψ as

$$\Psi : [M] \to X \tag{1a}$$

$$p \mapsto t(p)\psi(s(p)) + (1 - t(p))x_0, \tag{1b}$$

with s and t as defined above. Note that the choice of leaf for $p = p_0$ does not matter, due to $t(p_0) = 0$, i.e. $\Psi(p_0) = x_0$ in either case. The key deviation in this definition from that of Campen et al. [2016] is that we do not interpret $t(p)$ as radial coordinate in a polar system of a ball, but effectively as barycentric coordinate on a leaf of a natural foliation. Note that, in theory, the map could equivalently be defined as a map to a ball composed with a nonuniform radial scaling. This would involve calculations with irrational numbers though, defeating a key property of the foliation based approach: implementability with exact rational arithmetic for numerical safety. With our definition we can carry this over to our more general problem setting, as discussed further below.

Lemma 5.1. *The map Ψ, defined in* (1)*, is bijective and continuous.*

Proof. Let $p_1, p_2 \in [M]$ with $\Psi(p_1) = \Psi(p_2)$. For injectivity, we need to show $p_1 = p_2$. If $\Psi(p_1) = \Psi(p_2) = x_0$, then $t(p_1) = t(p_2) = 0$ because $x_0 \neq \psi(s) \in \partial X$ for all $s \in \partial[M]$, and therefore $p_1 = p_2 = p_0$. Else $t(p_1)\psi(s(p_1)) + (1-t(p_1))x_0 = t(p_2)\psi(s(p_2)) + (1-t(p_2))x_0 \neq x_0$, and therefore $\psi(s(p_1)) = \frac{t(p_2)}{t(p_1)}\psi(s(p_2)) + \left(1 - \frac{t(p_2)}{t(p_1)}\right)x_0$. This means, $\psi(s(p_1)) \in \partial X$ lies on the line segment spanned by x_0 and $\psi(s(p_2)) \in \partial X$. As this line segment intersects ∂X only once by star-shapedness, $\psi(s(p_1)) = \psi(s(p_2))$, and therefore $t(p_1) = t(p_2)$ and, by bijectivity of ψ, $s(p_1) = s(p_2)$. We conclude $p_1 = p_2$. Hence, Ψ is *injective*.

To show surjectivity, let $x \in X$. For $x = x_0$, we have $\Psi(p_0) = x_0$. For $x \neq x_0$ let $x' \in \partial X$ the intersection of the ray from x_0 through x with ∂X, unique by star-shapedness of X. Let $t = \|x - x_0\| / \|x' - x_0\|$ and $s = \psi^{-1}(x')$. This inverse exists because ψ is bijective. Then, the point p at relative position t on the leaf of $\mathcal{F}_M$ that contains surface point s is a point for which $\Psi(p) = x$. Hence, Ψ is *surjective*. Since Ψ is injective and surjective, it is *bijective*.

Regarding continuity, Campen et al. [2016] show that s and t are continuous. The prescribed surface map ψ is continuous by assumption. The map Ψ, consisting of compositions, products, and sums of these continuous functions, hence is *continuous*. □

5.1.1 Map Computation. After constructing the foliation $\mathcal{F}_M$ (or rather the foliation-defining direction field d, using Algorithm 1), the map Ψ at a point p can be evaluated by first tracing the leaf through p to determine $s(p)$ and $t(p)$, just as described in [Campen et al. 2016], and then evaluating (1). Also the inverse Ψ^{-1} can easily be evaluated: For a point $x \in X$, project it onto $x' \in \partial X$ along the ray from x_0 through x, and let t be the relative distance of this projection. Let $s = \psi^{-1}(x')$ and trace the leaf of $\mathcal{F}_M$ starting at s until the center p_0. Then determine the point on this leaf that lies at relative length t. This point is $p = \Psi^{-1}(x)$.

Numerics. The above constructions (as well as those in the following) were designed to rely exclusively on basic arithmetic operators. Calculations therefore remain within the rational numbers $\mathbb{Q}$. This allows implementing the algorithm using exact arithmetic based on rational number types, precluding the interference of numerical inaccuracies with the algorithm's correctness. Of course, if one truncates the output map to, e.g., standard floating point numbers, tiny inversions on the machine precision level may emerge nevertheless.

Distortion. Note that the main goal of the algorithm is to establish a map such that the binary criterion of bijectivity is reliably satisfied. For the consideration of additional soft objectives such as minimal distortion there are various options, from controlling the degrees of freedom of the foliation construction over bijectivity maintaining map optimization starting from the established map (cf. Section 2.2) as discussed by Campen et al. [2016]. This aspect, largely orthogonal to the bijectivity question, is not a focus of the present work.

Domain Class. Note that the approach could be applied beyond the class of star-shaped domains as well. In essence, instead of a natural foliation of a star-shaped domain X, one could use a foliation computed, analogously to $\mathcal{F}_M$, on a tetrahedral mesh of a general domain X. Definition (1) would be updated to map onto the leaves of that foliation instead of onto the straight line segments of the natural foliation of a star-shaped X. A key reason for restricting to star-shaped domains herein is that this enables obtaining a piecewise-linear version of Ψ more easily, exploiting the straightness of the leaves in X, as detailed in the following.

5.2 Piecewise Linearization

Depending on the application context it may be of benefit or even required to have a piecewise linear map, a map that can be represented by means of per-vertex map coordinates on a tetrahedral mesh (M, or a refined version thereof). The map Ψ does not possess this property, it is not linear per tetrahedron. We can easily obtain a piecewise linear approximation by evaluating Ψ at the vertices of M and defining a map through linear interpolation of these vertex map coordinates over the tetrahedra. Unfortunately, this approximation, while maintaining continuity, may break bijectivity.

For the case of a foliation based map to the unit ball domain, Campen et al. [2016] show that there is a refined version M' of mesh M, such that the above per-tetrahedron linearization of Ψ on M' yields an approximation that maintains bijectivity. The refinement is of nested kind, i.e. for each tetrahedron of M there is a set of one or more tetrahedra in M' that together occupy the same space. This keeps the relation to M simple. In appendix A we show that

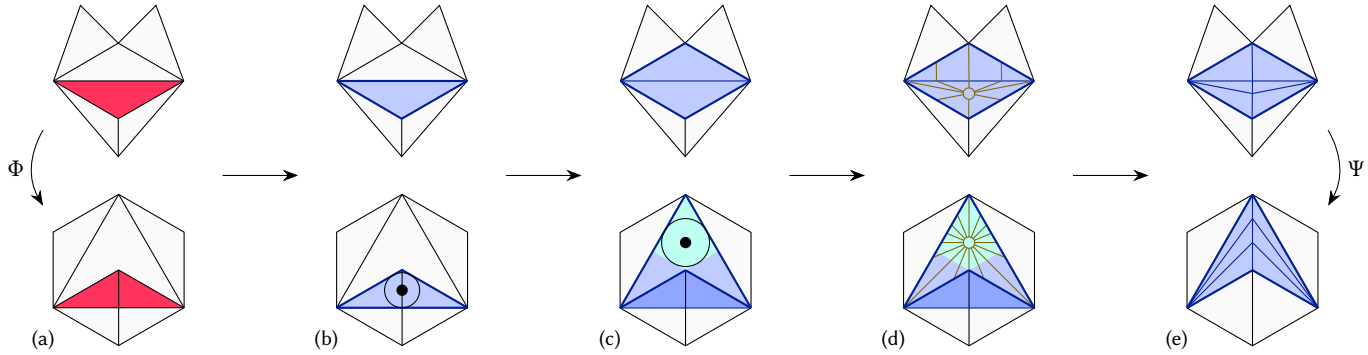


Fig. 6. Overview of our method. (a) An imperfect initial map Φ inverts one element (red). (b) A star (blue) starts growing at the inverted element. (c) The star finished growing, the guard is the insphere center of the kernel (cyan). (d) Inside the star, a bijective replacement map is computed using foliations. (e) After local refinement, the maps in combination form the desired global bijection Ψ.

this result applies to our setting as well and provide implementation details regarding the construction of the refined mesh.

Necessity of Refinement. Any kind of modification of the original mesh structure, whether by nested refinement or of different kind, may seem like a downside, as it may increase efforts to work with the resulting map in application contexts. Note, however, that refinement can be inevitable, depending on the interplay of the mesh structure of M, the shape of X, and the nature of ψ, when restricting to piecewise linear maps, cf. Fig. 7. Hence, methods that keep the mesh structure fixed can never be fully reliable in achieving bijectivity. In this sense the ability to systematically refine is a positive feature of the foliation based method. Of course it is desirable to keep refinement to a minimum. In this regard the described refinement procedure has significant room for improvement, posing interesting challenges for future work. As a great side effect of our main method described in the following Section 6, making use of the above described mapping algorithm in a local manner only, the amount of (superfluous) refinement is already heavily reduced.

6 GALAXY OF STARS

Assume a continuous map Φ of M that is linear per tetrahedron is given, such that its surface restriction $\Phi|_{[\partial M]}$ is bijective (and orientation-preserving) onto domain boundary ∂X. In the interior, the map may be noninjective, containing degeneracies and inversions. Such maps can easily be obtained using a variety of methods,

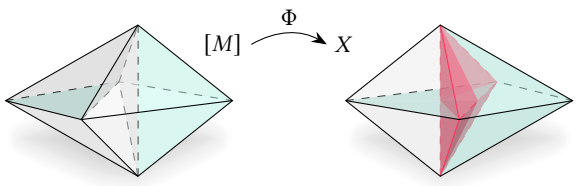


Fig. 7. Simple example of a mesh with five tetrahedra (left) with a prescribed boundary map (right) that does not permit a piecewise linear bijection (unless the mesh is refined); the inversion (red) is inevitable. Note that of the two inner triangles (cyan), one is horizontal, one vertical.

for example by a simple convex combination mapping, also called 3D Tutte embedding [Floater and Pham-Trong 2006].

Let $c \in C$ be a tetrahedron and v_i for $i \in \{1, 2, 3, 4\}$ its vertices, with index ordering chosen such that

$$\mathrm{vol}(c) = \frac{1}{6} \det \left(v_1 - v_4, \quad v_2 - v_4, \quad v_3 - v_4 \right) > 0.$$

We say that c is *inverted* by Φ if its image volume is non-positive:

$$\mathrm{vol}_\Phi(c) := \frac{1}{6} \det \left(\Phi(v_1) - \Phi(v_4), \Phi(v_2) - \Phi(v_4), \Phi(v_3) - \Phi(v_4) \right) \leq 0.$$

The map Φ is bijective *iff* it inverts no tetrahedra [Aigerman and Lipman 2013, Thm. 3]. In non-bijective maps computed by methods that (implicitly or explicitly) aim for bijectivity but do not guarantee it, inversions are often sparsely distributed local effects. The central idea of our method is to identify parts of the map that encompass these defects of Φ, and per part apply the algorithm from Section 5 to compute a bijective replacement map that can be seamlessly integrated into Φ. In the end, the map is globally bijective. The parts need to be chosen such that their image boundary is star-shaped for the algorithm to be applicable. We therefore call them *stars* in the following. The set of all stars required forms a *galaxy*. Fig. 6 gives an overview of the method.

In more detail, a star S is a submesh of M satisfying the following *star conditions*, where $\phi_S := \Phi|_{[\partial S]}$ denotes its boundary map:

- $[S]$ has ball topology,
- $\phi_S([\partial S])$ bounds a star-shaped set,
- ϕ_S is injective, thus bijective onto $\phi_S([\partial S])$,
- ϕ_S is orientation-preserving.

Then, mesh S together with boundary map ϕ_S form a valid input instance for the mapping algorithm from Section 5. Our goal therefore is to determine a galaxy $\mathcal{G}$ such that all tetrahedra inverted by Φ are contained in its stars.

A galaxy of minimal size (in terms of the number of tetrahedra contained in its stars) would be ideal for reasons of efficiency. However, even for a single inverted tetrahedron, no polynomial-time algorithm is known to determine its smallest star-shaped superset of tetrahedra. We therefore employ an efficient greedy strategy, incrementally growing stars starting from seeds.

6.1 Growing Stars

Conceptually, starting from a single tetrahedron c (inverted by Φ) we grow a star S^* by incrementally conquering adjacent tetrahedra one-by-one until the star conditions are satisfied (Section 6.1.1). For the selection of the next tetrahedron c^* to be conquered we employ a heuristic (Section 6.1.2). Note that this may also include inverted tetrahedra, i.e. one star can cover multiple of these. This is repeated, spawning further stars where necessary, until all inverted tetrahedra are covered. Algorithm 2 summarizes the overall procedure.

Algorithm 2: Galaxy Computation

Input: Tetrahedral mesh M with (possibly non-bijective) map Φ, linear per tetrahedron
Output: Galaxy $\mathcal{G}$

```
G = ∅
for each inverted tetrahedron c ∈ C
    if c ∉ S for all S ∈ G                    ▷ c not yet covered
        S* = ⟨{c}⟩                            ▷ spawn star at seed c
        do                                    ▷ growing
            Choose next tetrahedron c* by heuristic
            S* = ⟨S* ∪ {c*}⟩
            if c* ∈ S for any S ∈ G           ▷ star collision
                S* = ⟨S* ∪ S⟩                 ▷ absorb other star
                G = G \ {S}
        while S* violates the star conditions
        G = G ∪ {S*}
return G
```

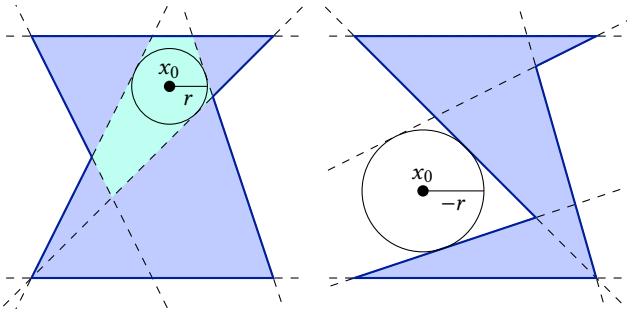
Termination. Assuming the domain X of Φ is star-shaped, star growing will always terminate with success: In the worst case, a star will grow to $S^* = M$, which satisfies the star conditions. In this case, with $\mathcal{G} = \{M\}$, the method effectively degenerates to a direct global application of the mapping algorithm from Section 5.

Star Collisions. If a growing star S^* collides with another star $S \in \mathcal{G}$, in the sense that the to be conquered c^* is already included in S, we let S^* absorb S by setting $S^* = \langle S^* \cup S \rangle$ and $\mathcal{G} = \mathcal{G} \setminus \{S\}$. This ensures that in the end no two stars overlap, i.e. $S_1 \cap S_2$ does not contain any tetrahedra for $S_1, S_2 \in \mathcal{G}$, $S_1 \neq S_2$. Note, however, that they may touch, i.e. may share vertices, edges, or triangles.

Inversion Clusters. Note that a star's boundary cannot contain the face between two face-adjacent inverted tetrahedra; the map Φ is not orientation-preserving on that face, in violation of the star conditions. For efficiency, we therefore conquer entire connected components of inverted tetrahedra at once. For clarity, however, Algorithm 2 lists the simple one-by-one version.

6.1.1 Checking Star Conditions. On the boundary of the growing star we perform an oriented star-shapedness check and an injectivity check. A separate check for ball topology is not necessary; it is implied by star-shapedness.

Star-Shapedness Check. To certify star-shapedness, we compute a witness guard point x_0 inside the kernel (Section 4.3). More specifically, we can compute the Chebyshev center [Boyd and Vandenberghe 2004, §8.5], the center of the largest inscribed sphere, of the kernel using a linear program (LP) as follows:



(a) Star-shaped. (b) Non-star-shaped.

Fig. 8. Star-shapedness can be tested and in parallel the kernel's Chebyshev center x_0 be determined by computing the maximum insphere radius r.

$$\max_{x_0, r} \quad r \tag{2a}$$

$$\text{s.t.} \quad {n_f}^\mathsf{T} x_0 + r \leq {n_f}^\mathsf{T} a_f \quad \text{for all triangles } f \in \partial S^*. \tag{2b}$$

In this r is an auxiliary variable that effectively measures the radius of the insphere, n_f denotes the image normal of triangle f, and a_f an arbitrary point contained in its image. See Fig. 8 for an illustration. For the case of a degenerate triangle image $\Phi(f)$, let $n_f = 0$.

Regarding the orientation of the normals n_f, let v_i for $i \in \{1, 2, 3\}$ be the vertices of f, indexed such that $(v_2 - v_1) \times (v_3 - v_1)$ points outwards of S^* at f. The normals for the LP are computed as

$$n_f = \frac{(\Phi(v_2) - \Phi(v_1)) \times (\Phi(v_3) - \Phi(v_1))}{\|(\Phi(v_2) - \Phi(v_1)) \times (\Phi(v_3) - \Phi(v_1))\|}. \tag{3}$$

After solving the LP, if $r > 0$ we say the star-shapedness check succeeded; the resulting point x_0 lies on the inside with respect to all boundary triangles' supporting planes in image space. If in addition the boundary is simple (i.e. does not self-intersect) this means that it bounds a region of $\mathbb{R}^3$ that is star-shaped and x_0 is the center of its kernel's insphere. We note that it is possible (and in some cases faster) to calculate an explicit geometric representation of the kernel without the use of an LP [Sorgente et al. 2022]. However, in case of non-star-shapedness, i.e. $r \leq 0$, the point x_0 resulting from the LP contains valuable information that we can leverage for the growth heuristic (Section 6.1.2).

Injectivity Check. The above check verifies that the star boundary forms a star-shaped set in image space and that its orientation is proper. It does, however, not yet catch cases where the boundary map ϕ_S is non-injective. Fig. 9 shows a 2D example of a boundary image winding around a central point multiple times; it passes the star-shapedness check, but the boundary image self-intersects, i.e. the boundary map is non-injective.

Therefore, if the star-shapedness check succeeds, an additional check for intersections between pairs of image triangles in $\phi_S([\partial S])$ needs to be performed. If an intersection is detected, the star keeps growing. In our extensive experiments (cf. Section 7) this case actually never occurred. This is likely connected to the above mentioned atomic treatment of inversion clusters (which, as a side effect, precludes configurations like in Fig. 9).

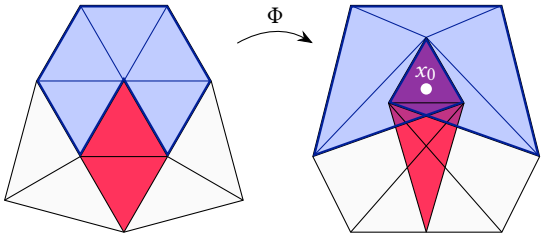


Fig. 9. 2D illustration of a self-intersecting star boundary image (right, bold), winding around x_0 twice. The two triangles marked in red are inverted by Φ.

6.1.2 Growth Heuristic. The next tetrahedron c^* to be conquered for a growing star S^* is chosen among the tetrahedra that are face-adjacent to S^*, so as to maintain a connected set. Among these we would like to select one that helps towards reaching a state that satisfies the star conditions. Among the triangles of ∂S^* we compute $f^* = \arg\max_f {n_f}^\mathsf{T} x_0 - {n_f}^\mathsf{T} a_f$, with n_f and a_f as before, and select the tetrahedron $c^* \notin S^*$ as the one incident to f^*. Fig. 10 illustrates this choice. The rationale for this choice is as follows.

As discussed above, the main reason for violation of the star conditions is non-star-shapedness, as indicated by a nonpositive radius r in the solution of (2). The triangle f^* is one of the triangles for which the constraint (2b) is sharp, i.e. whose supporting plane has the highest signed distance (namely $-r$) to the pseudocenter x_0 of S^*. In this sense this triangle is a main reason for the radius r not being larger than it currently is. By conquering the incident c^*, f^* no longer is a boundary triangle of $\langle S^* \cup \{c^*\}\rangle$.

Note that this is a greedy heuristic that does not guarantee to ultimately find the smallest star; the smallest star might not contain c^* and have a center from which f^* is seen from the correct side.

Remark. Let us point out that the image of a finished star S under Φ may (due to inversions) lie partially outside of the star-shaped boundary $\phi_S([\partial S])$ and even outside of the overall domain boundary ∂X. This is not a problem because only the boundary image $\phi_S([\partial S])$ is made use of in our method, the map of the interior is replaced using the algorithm from Section 5.

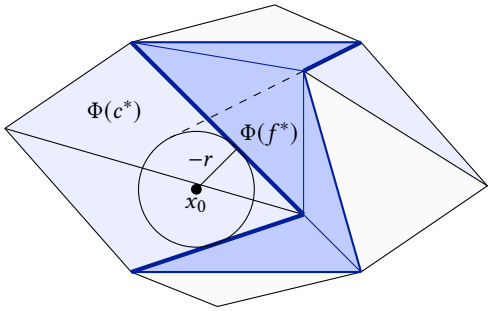


Fig. 10. Selection of the next tetrahedron c^* to be added to a growing star based on its pseudo-center x_0 in image space. Note that the "most violating" element f^* (a face in 3D, an edge in this 2D illustration) is not unique: The three bold blue elements have the same distance to x_0; one of these is chosen arbitrarily. The incident tetrahedron then is chosen as c^*.

6.2 Star Map Fusion

Once the galaxy $\mathcal{G}$ is computed, per star $S_i \in \mathcal{G}$ with boundary map ϕ_{S_i} a new map Ψ_i can be computed using the algorithm from Section 5. Together with the (non-bijective) map Φ we can compose these to the global bijective map Ψ:

$$\Psi(p) = \begin{cases} \Psi_i(p) & \text{if } p \in [S_i] \\ \Phi(p) & \text{otherwise.} \end{cases} \tag{4}$$

This map can be evaluated, at a given point $p \in [M]$, as follows. Check whether p lies inside any star. If not, evaluate the (piecewise linear) map Φ at p and return the result. Otherwise, if p lies inside star S_i, evaluate the foliation based map Ψ_i at p (by tracing the corresponding leaf, etc., as described in Section 5.1.1).

For many practical purposes it will be more useful, however, to have a unified global piecewise linear map available. To this end we can linearize each star map Ψ_i as described in Section 5.2, and combine these with the map Φ on $M \setminus \bigcup_i S_i$. Due to the fact that the linearization implies some refinement for the meshes S_i, this combination requires a little extra effort to yield a conforming mesh again at the interface of the stars. This is described in the following.

6.2.1 Star Linking. The transition between each Ψ_i and Φ is continuous because the latter was used to constrain the surface map of each Ψ_i. But when refining S_i to S_i' to yield a piecewise linear version Ψ_i' of Ψ_i (cf. Section 5.2), the tetrahedral meshes S_i' and $\langle M \setminus S_i\rangle$ do not fit together conformingly anymore; multiple tetrahedra of S_i' may be adjacent to one tetrahedron of M via a face or an edge. We therefore refine one layer $O = \langle\{c \in M : c \notin S_i \wedge c \cap [S_i] \neq \emptyset\}\rangle$ of tetrahedra around S_i in M, so as to conformingly link S_i' and M.

Each tetrahedron $c \in O$ is refined as follows. Let $s = c \cap [S_i]$.

- If s consists of vertices only, c does not need to be refined.
- If s consists of an edge e (and possibly further vertices), c can be refined as a *fan* of tetrahedra: In S', e was possibly split into multiple subedges; for each of these a tetrahedron is spanned with the opposite edge of e in c (Fig. 11a).
- If s consists of a triangle f (and possibly its opposite vertex), c can be refined as a *bouquet* of tetrahedra: In S', f was possibly split into multiple subtriangles; for each of these a tetrahedron is spanned with the opposite vertex of f in c (Fig. 11b).

Besides these single-edge and single-face cases, c can be adjacent to S_i in more complex ways. To avoid a long list of special refinement rules for any adjacency pattern, we first split such tetrahedra, reducing to the above simple cases.

(a) Fan. (b) Bouquet.

Fig. 11. Tetrahedra in the interface layer surrounding a star are refined to either a fan or a bouquet of subtetrahedra, so as to yield conformance to the refined star, adjacent via the (a) edge or (b) face marked in blue.

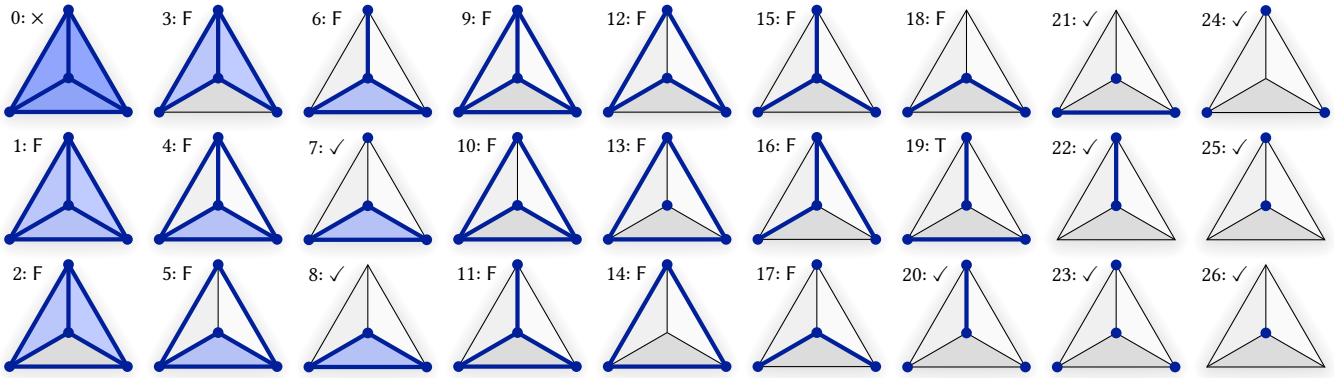


Fig. 12. Configurations of the set s (blue) on a tetrahedron c. Case 0 cannot occur in our setting due to star-shapedness (×). Some cases need no refinement or can easily be handled by refining them as a fan or bouquet (✓). All others can be reduced to these simple cases via a tetrahedron split (T) or face splits (F).

Concretely, if s consists of two non-adjacent (i.e. opposite) edges of c (case 19 in Fig. 12), we perform a barycentric tetrahedron split, splitting c into four tetrahedra, inserting a new vertex in its barycenter. Otherwise, we perform a barycentric face split on every face of c that is not in s but more than one of its edges is. Fig. 12 shows all potential configurations for s in c. One easily verifies that these split rules leave only simple tetrahedra, with s a single simplex (with vertices), in each of these cases.

The global map is identically carried over onto the subtetrahedra created by splitting and by fan or bouquet refinement, simply by linearly interpolating map values onto the new vertices. Algorithm 3 summarizes the complete method.

Algorithm 3: Overall Method

Input: Tetrahedral mesh M with (possibly non-bijective) map Φ, linear per tetrahedron
Output: Refined tetrahedral mesh M' with bijective map Ψ, linear per tetrahedron

$M' = M$
$\Psi = \Phi$
Compute galaxy $\mathcal{G}$ ▷ Algorithm 2
for each star $S_i \in \mathcal{G}$
  Compute map Ψ_i on S_i with $\psi = \Psi|_{[\partial S_i]}$ ▷ Algorithm 1
  Linearize Ψ_i to Ψ_i' refining S_i to S_i' ▷ Section 5.2
  $O = \langle \{c \in M' : c \notin S_i \wedge c \cap [S_i] \neq \emptyset\} \rangle$
  $M' = \langle M' \setminus (S_i \cup O) \rangle$
  Refine O to O' to match S_i' maintaining Ψ ▷ Section 6.2
  $M' = M' \cup O' \cup S_i'$ with $\Psi|_{[S_i']} = \Psi_i'$
return M'

6.3 Numerics

Like in the algorithm from Section 5, all calculations in the proposed galaxy based method occur in $\mathbb{Q}$—with one exception: setting up the LP (2) requires the calculation of vector lengths for normalization in (3). While this normalization could be skipped or approximated without affecting the sign of r, for efficiency we can even solve the LP approximately using standard floating point arithmetic, and then check the resulting point x_0 for being a guard using exact orientation predicates. Formally, this yields a test for star-shapedness that is conservative, which is sufficient for correctness.

7 RESULTS

We implemented the described method in C++, using GMP for its rational number type, and CGAL for 2D arrangements (refinement patterns) and to solve the Chebyshev center LP (2). All experiments were conducted on a system with AMD EPYC 7742 processor.

7.1 Datasets

We make use of the following datasets with problem instances for purposes of evaluation and comparison.

- TLC: The 46 volumetric instances with a star-shaped domain from the mapping dataset[1] of Du et al. [2020].
- CUB: The 60 instances from the mapping dataset[2] of Brückler et al. [2022b], with prescribed boundary maps onto cuboid domains.

To enable a broader evaluation, we furthermore create new mapping problem instances, bringing the total number of instances to over 7500. From all the surface meshes from the Thingi10K dataset [Zhou and Jacobson 2016] that have genus 0, tetrahedral meshes are generated using TetWild [Hu et al. 2018]. To increase diversity, for a 10 % subset of this set of surface meshes we additionally generate tetrahedral meshes using TetGen [Si 2015]. Using a simple randomized procedure, we attempt to generate example bijective boundary maps for all these, so as to yield input instances for testing. As target domains we use a sphere, a cube, and a nonconvex domain (a star-shaped deformed bipyramid). In brief, by picking random boundary vertices and connecting them by shortest paths, the procedure attempts to partition the surface, structurally compatible with the cube, the bipyramid, or in the case of the sphere two disks. Each patch of the partition is then mapped onto its corresponding face of the domain using a 2D Tutte embedding (in case of the sphere: a disk that is then projected onto one of the hemispheres). If unsuccessful (e.g. shortest paths overlap, invalidating the partition), the

[1] https://github.com/duxingyi-charles/Locally-Injective-Mappings-Benchmark
[2] https://github.com/HendrikBrueckler/CuboidBlocks

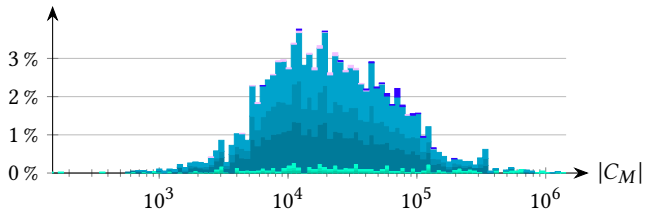


Fig. 13. Distribution of mesh size (number of tetrahedra) for each dataset.

model-domain combination in question is dropped, otherwise it is added to the datasets of problem instances—unless it is so simple that already a 3D Tutte embedding yields a bijective result. This led to the following additional datasets:

- TWS: 2738 TetWild meshes mapped to spheres
- TWC: 1884 TetWild meshes mapped to cubes
- TWN: 2340 TetWild meshes mapped to nonconvex domain
- TGS: 249 TetGen meshes mapped to spheres
- TGC: 108 TetGen meshes mapped to cubes
- TGN: 166 TetGen meshes mapped to nonconvex domain

The distribution of mesh sizes of all datasets is shown in Fig. 13. The whole collection of these six additional datasets is available at https://github.com/SteffenHinderink/GalaxyMaps.

7.2 Evaluation

We run our galaxy based method to generate a bijective map Ψ following (4) on all mapping problem instances. The initial maps are generated simply using 3D Tutte embedding. As can be seen in Table 1, as expected, it succeeds to produce a volumetric bijection for all instances. Fig. 14 shows an example instance in which the initial map even maps partly outside of the prescribed domain boundary, a case relatively common for Tutte embeddings to nonconvex domains. This makes no difference to our method; the map is modified into a bijective map into the intended domain.

7.2.1 Star Statistics. To gain further insight into our method we analyze the galaxies. Fig. 15 shows the distributions of the number of stars per galaxy. Fig. 16a further examines the size of the individual

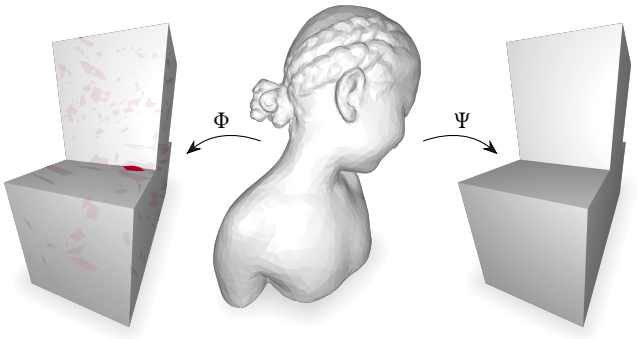


Fig. 14. The image of the mesh under the non-bijective input map Φ (left) lies partially outside of ∂X (at a concave region). Our method produces a bijective map Ψ, therefore the image lies properly inside (right).

Table 1. Number of successes and success rate to produce a bijective map. The initial 3D Tutte embedding fails for all instances in the datasets, i.e. our method has work to do in every case. It reliably produces a bijection.

Dataset	Total	Tutte	Tutte rate	Ours	Ours rate
TLC	46	0	0 %	46	100 %
CUB	60	0	0 %	60	100 %
TWS	2738	0	0 %	2738	100 %
TWC	1884	0	0 %	1884	100 %
TWN	2340	0	0 %	2340	100 %
TGS	249	0	0 %	249	100 %
TGC	108	0	0 %	108	100 %
TGN	166	0	0 %	166	100 %

stars. Compared to the whole meshes, the stars are typically very small, the vast majority not even containing more than 20 tetrahedra. This means that our method succeeds in localizing small regions around the inverted tetrahedra, where the initial map can be fixed based on small local mapping problems. Notice that the tetrahedral meshes generated by TetGen seem to have characteristics that are particularly favourable in this regard. Fig. 16b shows how many of the tetrahedra of a mesh are part of any star, i.e. of the galaxy.

Alternative Galaxies. Let us remark that the galaxies could also be created in different ways than proposed, using different classes of target shapes when growing the stars. First, instead of treating the guard point x_0 as a variable in every step while growing a star S^*, it could be fixed, e.g. at the center of the seed tetrahedron. This

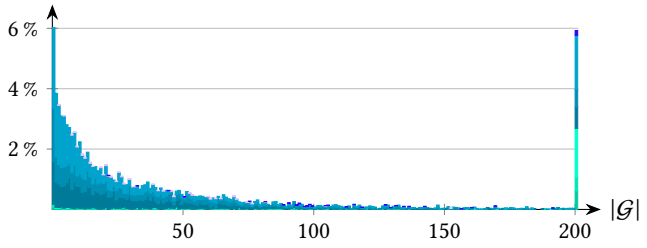
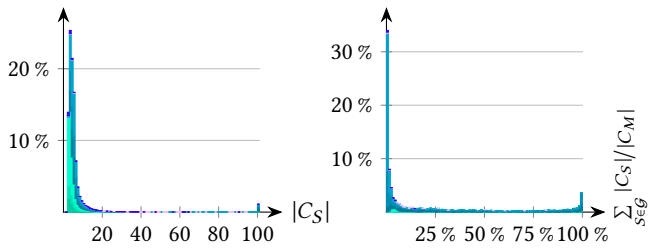
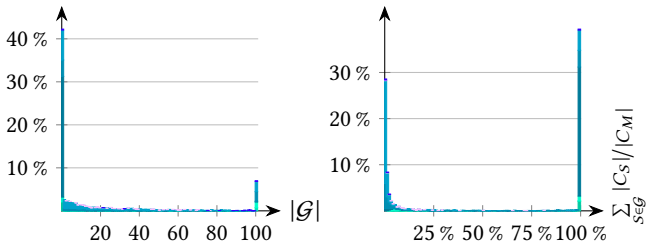


Fig. 15. Distribution of galaxy size (number of stars), over all dataset instances. The last bar covers all sizes > 200.



(a) Number of tetrahedra per star. (b) Ratio of tetrahedra in galaxy.

Fig. 16. Distribution of star size and galaxy size (number of contained tetrahedra) relative to the whole mesh M, over all dataset instances.



(a) Number of stars per galaxy. (b) Ratio of tetrahedra in galaxy.

Fig. 17. Distribution of galaxy size when using alternative *fixed guards*.

would remove the need to solve the LP (2) and thereby speed up the process. Second, convex instead of star-shaped sets could be used since they are a subset of the latter. Both options have the benefit of allowing for the determination of the actual *minimum* such set of tetrahedra enclosing a seed, by means of a greedy procedure that simply adds any violating tetrahedron to S^* until there is none left.

However, our experiments reveal that the downsides due to the more restrictive nature of these alternatives outweigh their benefits. While many stars grown with a fixed guard have a similar size as stars grown with a free guard, for others the desired star-shaped state with respect to the fixed guard cannot be reached (in which case we used the fallback to $S^* = M$). That is why many instances have only 1 star in their fixed-guard galaxy, cf. Fig. 17a—it often is a star equal to all of M; this is the case for 39.14 % of all instances, as reflected in the tall last bar in Fig. 17b. This problem is only exacerbated by using convex sets: For no instance multiple small sets resulted, all such galaxies contain 1 global convex set, the size of which typically is very similar to M; at most some tetrahedra along the boundary are sometimes excluded.

7.2.2 Runtime. Next, we evaluate the runtime of our method. Fig. 18 shows the distribution of the total runtime. Its varies strongly, between milliseconds and days. While a bulk of instances is bijectively mapped in seconds, especially complex meshes mapped to non-convex domains can take much longer. In particular, 85 instances (i.e. ≈ 1 %) take longer than one day to be parametrized. Characteristically, these are all very large instances ($|C| > 150\,000$) from the

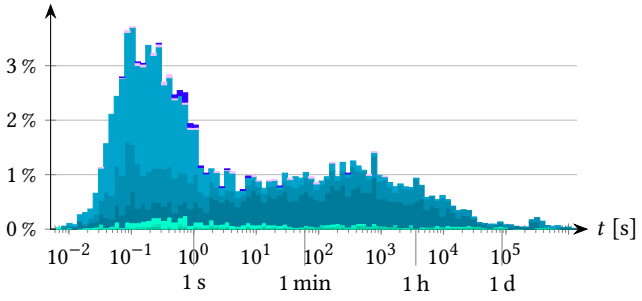


Fig. 18. Distribution of runtime over all dataset instances. A bulk is handled within seconds, but especially instances with non-convex domains can take longer, in rare cases multiple hours.

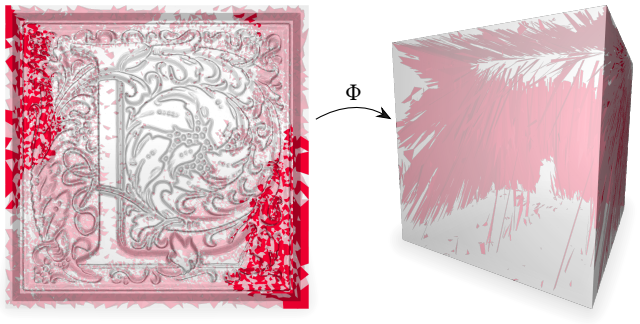
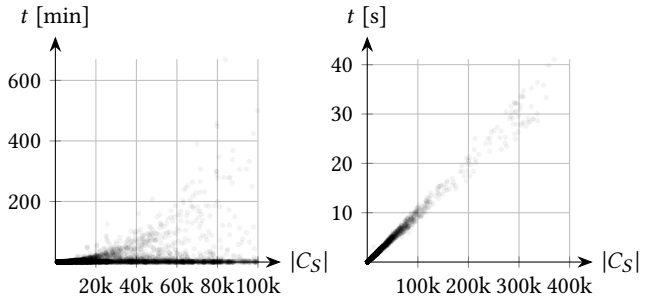


Fig. 19. Example of an instance (left) for which the initial Tutte map (right) has vast inversion clusters (red), making the localizing galaxy approach ineffective; here it leads to one star, containing all but 35 tetrahedra.

newly generated datasets and have bad initial maps with major clusters of inverted tetrahedra (see Fig. 19) resulting from an intricate object shape or a nonconvex domain shape. Together this implies the growth of large stars.

The size of the stars is the main factor for the total runtime. This is made evident by analyzing the runtimes of the different steps of the method. Fig. 20a shows how long it takes to grow a star. Note that many of the big stars arise from collisions of small stars (which are grown quickly); this is the reason for the many data points near the horizontal axis in the plot. But growing a large star from scratch takes longer per tetrahedron the bigger the star gets. This is the main bottleneck of our current implementation—re-solving the entire LP each time a tetrahedron is added to the growing star, and safely testing star-shapedness using (unfiltered) exact arithmetic. The time for remapping a star, by in particular generating the direction field that defines a foliation of the star, likewise depends on the star's size. But compared to star growing, this step is rather negligible, cf. Fig. 20b. It only makes up 0.01 % of the total runtime for the above 85 long-running instances.

7.2.3 Linearization. We now turn to furthermore refining the stars, so as to obtain a piecewise linear version of Ψ as described in Section 5.2. As also discussed in [Campen et al. 2016, §7], this process is expensive. While a portion of this refinement is superfluous and



(a) Growing. (b) Foliation.

Fig. 20. Growing time (a) per star and time to compute the foliation-based map (b) per star, in relation to the (final) star size $|C_S|$.

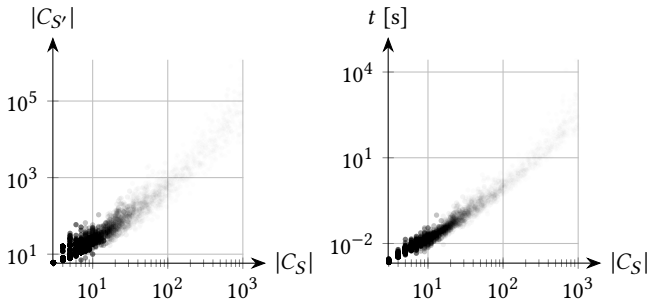
Table 2. Statistics regarding piecewise linearization (PL) through mesh refinement. It succeeded in obtaining a bijective piecewise linear map version for all instances we evaluated it on.

Dataset	Total	All stars ≤ 1000	Ratio	PL success
TLC	46	43	93.48 %	43
CUB	60	43	71.67 %	43
TWS	2738	2511	91.71 %	2511
TWC	1884	1064	56.48 %	1064
TWN	2340	626	26.75 %	626
TGS	249	210	84.34 %	210
TGC	108	54	50.00 %	54
TGN	166	46	27.71 %	46

could be decimated again in the end, the high intermediate size and runtime limits practical global applicability of this linearization approach to meshes with a few hundred or perhaps thousand tetrahedra. Experiments with our implementation, applied globally, confirm this. The number of tetrahedra may increase by orders of magnitude due to refinement and the runtime to perform it behaves accordingly, cf. Fig. 21. If applied globally, without our galaxy based approach, it would take more than a day for more than 90 % of the instances of our datasets.

In our method, due to the localized application of the algorithm, per star of the galaxy, this issue is strongly ameliorated. As seen in Section 7.2.1, most of the stars stay relatively small, the vast majority contain less than 20 tetrahedra, permitting swift refinement for linearization.

For experimental practicability, we restrict our analysis of this linearization procedure in the context of our method to those input instances whose galaxies do not contain a star larger than 1000 tetrahedra. Table 2 reports for what percentage of instances that is the case. On all resulting maps of the experiments with piecewise linearization through refinement, we performed a test, checking for tetrahedra that are inverted or degenerated by the map. It verified bijectivity in every case, as reported in the last column of Table 2. The global refinement ratio is shown in Fig. 22. Comparing this to the typical ratios observed in Fig. 21a, the major benefit of the



(a) Number of tetrahedra. (b) Time.

Fig. 21. Linearization: Number of resulting tetrahedra (a) and time (b) per star to perform the mesh refinement, depending on the star's size.

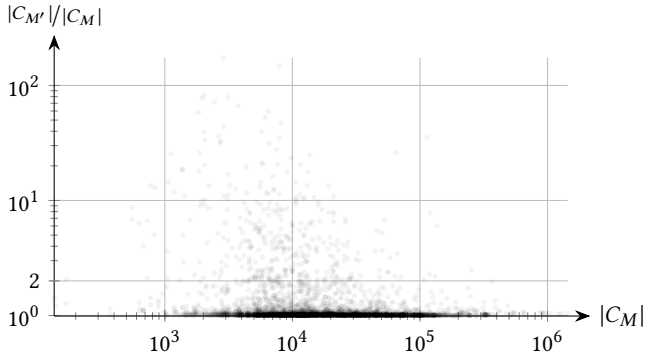


Fig. 22. Global refinement ratio due to piecewise linearization, depending on input mesh size (over a subset of test instances, with stars no larger than 1000 tetrahedra, cf. Table 2). The majority (92.04 %) has a ratio below 2.

localizing galaxy approach in terms of heavily reduced refinement becomes clear. The overall runtime of the method, including final linearization through mesh refinement, is reported in Fig. 23.

7.2.4 Non-Star-Shaped Domains. As a brief remark, our method also succeeds on 310 of the 858 volumetric instances that have a *non*-star-shaped prescribed domain (thus actually violate our input assumption) from the mapping dataset of Du et al. [2020], even though they are beyond the scope of our method. Of those, 282 instances have stars no larger than 1000 tetrahedra. Statistics and an example of these are depicted in Fig. 24. This is possible as long as each star converges to a ball-topology star-shaped part of the domain.

7.3 Comparison

In the following we compare our method to two recent methods, TLC [Du et al. 2020] and FFM [Garanzha et al. 2021], that aim for bijective maps via an optimization approach rather than constructively. Let us remark that their problem setting is not identical to ours, which should be taken into account when interpreting the results. In particular they assume a fixed mesh relative to which the output map is supposed to be piecewise linear. On the one hand this

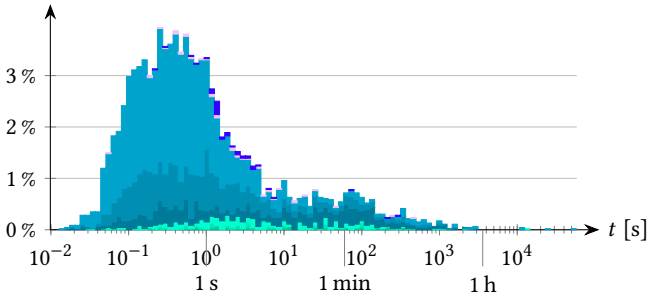


Fig. 23. Distribution (over a subset of test instances, with stars no larger than 1000 tetrahedra, cf. Table 2) of the total time to determine the galaxy, foliate all stars, perform piecewise linearization of all stars through refinement, and fuse them with the remainder of the global map.

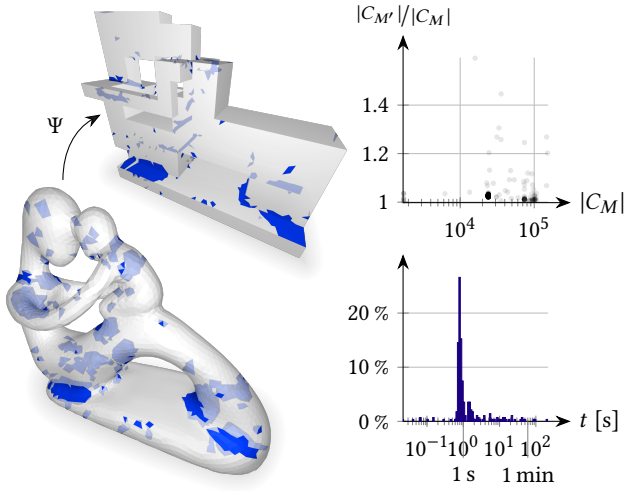


Fig. 24. Our method may succeed even when the mapping domain is not star-shaped. In this example (left, with a surface of genus 4) it is not even of ball topology—but its stars (blue) are, which is sufficient. The plots show the refinement ratio and runtime distribution (average: 4 s) over the successful non-star-shaped instances (cf. Section 7.2.4), analogous to Figures 22 and 23. The peaks in the plots are due to one mesh occurring multiple times (with differing domain shape) in the dataset of Du et al. [2020].

clearly is beneficial for various applications, on the other hand the fixed discretization may be adverse to achieving bijectivity.

On that note, to aid these methods we preprocess the input meshes, so as to grant more degrees of freedom: An interior edge $e \in M \setminus \partial M$ is split if each incident vertex $v \in \partial M$; an interior face $f \in M \setminus \partial M$ is split if each incident edge $e \in \partial M$. This is to avoid some cases in which these methods trivially could not succeed because they operate with a fixed mesh and have no means to systematically perform mesh refinement; the input mesh in its original form may not permit any valid (PL) map that respects the prescribed boundary map. Fig. 7 illustrates this. Note that our method makes no restricting assumptions on the input mesh's connectivity, i.e. such pre-refinement is not required.

7.3.1 Reliability. FFM and TLC are best-effort methods. While impressively high success rates have been observed on triangle meshes in 2D, they seem to struggle more often in the 3D case, depending on the problem instance characteristics. This can be observed in Table 3, which lists their success rates on the various datasets. Note that we used the implementations[3,4] provided by the respective authors for these experiments. While on the TLC dataset (as well as on the non-star-shaped instances, cf. Section 7.2.4) both methods have a success rate of 100 %, on all others it is significantly lower, between 30 % and 80 %.

Regarding the question whether the cases that are difficult for our method (in terms of runtime or amount of refinement) are correlated to the cases difficult for TLC or FFM we observed the following. The success rates of TLC and FFM on the subset of input instances

[3]https://github.com/duxingyi-charles/lifting_simplices_to_find_injectivity
[4]https://github.com/ssloy/invertible-maps

Table 3. Number of successes and success rates of TLC and FFM to produce a bijection, in contrast to our method's success rate reported in Table 1.

Dataset	Total	TLC	TLC rate	FFM	FFM rate
TLC	46	46	100 %	46	100 %
CUB	60	35	58.33 %	28	46.67 %
TWS	2738	1892	69.10 %	1547	56.50 %
TWC	1884	1499	79.56 %	1144	60.72 %
TWN	2340	1355	57.91 %	1213	51.84 %
TGS	249	82	32.93 %	97	38.96 %
TGC	108	48	44.44 %	46	42.59 %
TGN	166	50	30.12 %	51	30.72 %

that caused no stars larger than 1000 tetrahedra (cf. Table 2) show no peculiarity, they are similar to those on the full dataset. On the instances that took our method more than a day (cf. Section 7.2.2), however, TLC and FFM perform relatively badly, succeeding on only 9 and 0 out of these 85, respectively.

7.3.2 Runtime. Since TLC as well as FFM use numerical optimization to aim for bijectivity, their runtimes depend on the parameters used for their solvers. We use the provided default parameters. For FFM we added a check for NaNs after each iteration, outputting the best numerically sane map state so far when numerical limits are reached. Figures 25 and 26 show the runtimes of the methods

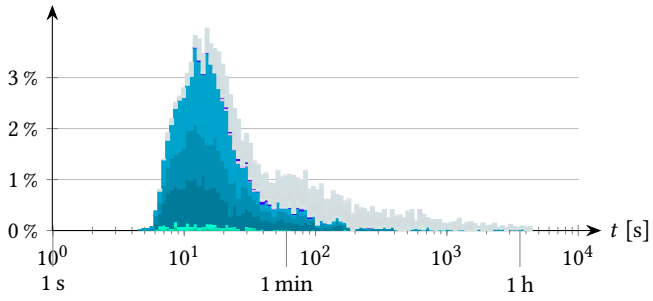


Fig. 25. Distribution of runtime of the TLC method on all dataset instances. Failure cases, with a non-bijective result, are greyed out.

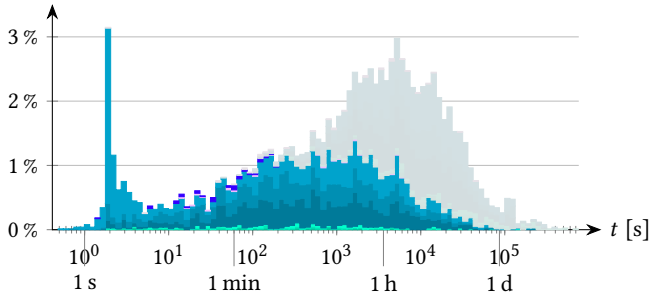


Fig. 26. Distribution of runtime of the FFM method on all dataset instances. Failure cases, with a non-bijective result, are greyed out.

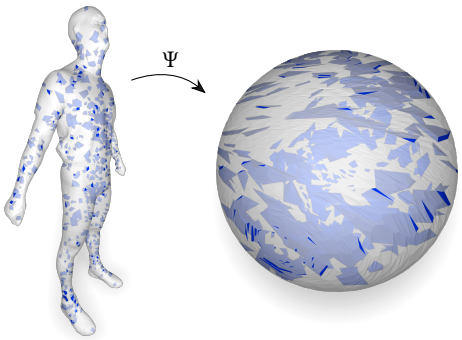


Fig. 27. An example of an instance that is successfully mapped, bijectively and piecewise linearly, by TLC, FFM, and our method. TLC and FFM each take about 2 min. Our method takes 5 s.

resulting from these parameters. While FFM exhibits similar runtimes of up to over 1 day as our method, TLC—when it succeeds—on average is relatively fast in comparison. This is not generally the case, though. There are instances on which TLC and FFM succeed and where our method (even including linearization) is significantly faster. One example for this is shown in Fig. 27. Also on the examples from Fig. 24 this is the case, with average runtimes of 25 s and 61 s, respectively, compared to 4 s.

7.3.3 Escalating Approach. TLC appears suitable to be used as an intermediate tier in an escalating approach because it is relatively fast and does not change mesh connectivity. Concretely, given a mapping problem instance, simple Tutte embedding is applied first. If the result is not bijective, TLC is applied to improve this map. If the result is still not bijective, our method comes to the rescue, finally turning this improved map into a bijection. The runtimes of this escalating 3-tiered approach are depicted in Fig. 28. In comparison to Fig. 18 it is apparent that, while the very quick instances take a bit longer with the intermediate TLC tier, the very long runs are prevented. A key reason is that TLC, even if unsuccessful in obtaining a bijection, often improves the inital map in the sense of

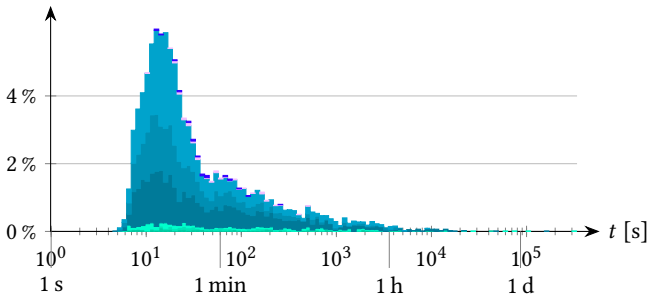


Fig. 28. Distribution of total runtime of the 3-tiered escalating approach (Tutte → TLC → GalaxyMaps) over all dataset instances. Notice that, in comparison to the standard approach (Tutte → GalaxyMaps) in Fig. 18, while the runtime of the very quick instances increases due to additional overhead, the runtime of the very long-running instances is reduced favourably in many cases.

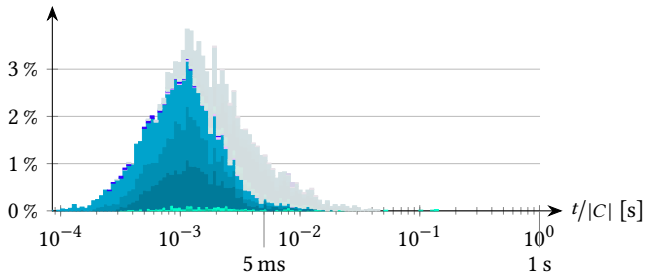


Fig. 29. Distribution of TLC's average runtime *per tetrahedron*.

in particular pulling the clusters of inverted tetrahedra that commonly arise from Tutte embedding at nonconvex regions, into the prescribed domain boundary and uninverting at least a significant portion of them. This leaves less work, with smaller stars, for our method.

There is potential to further improve this approach. For instance, a time limit could be imposed on the intermediate tier. According to Fig. 25, full success of TLC is unlikely once it has been optimizing for a few minutes. We also did not observe significant improvement of the maps, in terms of the number of inverted tetrahedra, beyond that point. Granting additional time thus seems rather futile. Ideally, of course, such a time limit should be relative to the size of the instance. As an optimization step of TLC takes time close to linear in the size of the mesh, a time budget per tetrahedron is reasonable. Based on Fig. 29, a budget somewhere around 5 ms per tetrahedron appears reasonable; beyond that, the chance of eventual failure is much higher than that of success. We leave further exploration of configurations and potential variants to future work.

7.3.4 Distortion. In this work our focus is on bijectivity, not the quality of the map beyond that, in terms of mapping distortion. This focus is reasonable in the sense that methods exist, discussed in Section 2.2, that assume as input a bijective map, which is then improved in terms of distortion while maintaining bijectivity. Our method essentially provides a valid starting point in this context.

To give an idea of the magnitude of distortion, Table 4 reports the minimum Jacobian determinant of Ψ (over all tetrahedra) for an example instance. It is strictly positive by construction with our method, but can be very close to zero. Methods such a TLC or FFM, which (indirectly) take distortion into account while aiming for bijectivitiy, yield maps of less distortion. As can be seen in the last column, applying a simple distortion optimization to our method's result can bring the map to a comparable level of distortion.

Table 4. Map distortion in terms of the minimum determinant of the map's Jacobian over all elements of an example mapping instance on which also TLC and FFM succeed (the Bimba instance from the TLC datatset).

	Tutte	Ours	FFM	TLC	Ours + optim.
$\min \det J$	-24.50	$2.07 \cdot 10^{-7}$	$1.00 \cdot 10^{-3}$	$1.27 \cdot 10^{-3}$	$1.45 \cdot 10^{-3}$

7.4 Limitations

By design, our method is restricted to star-shaped mapping domains. Due to the galaxy approach, it may well work for other domains, too, cf. Section 7.2.4, but in such cases it is without guarantee. While the class of star-shapes is already significantly larger than that of convex shapes, which various other methods are restricted to, a generalization to arbitrary domains would be an asset. Some divide-and-conquer approach or the direction pointed out in Section 8 could be followed to that end in future work.

As for practical limitations, the method can be slow. Note, however, that there is no faster alternative with bijectivity guarantee, and even state-of-the-art methods without guarantee but reasonable success rates are not necessarily faster (cf. Section 7.3). The two main factors, by far, are the star growing and the refinement for linearization. Fortunately, there is obvious potential for speed-up in these parts: Instead of solving the LP for star growing from scratch every time, warm starting could be employed, exploiting that only a few constraints change between iterations. Instead of solving the LP after each single tetrahedron added to a star, the guard point could be kept fixed for batches. Vertex images could be adjusted at the growing star's boundary to aid star-shapedness. Particularly attractive would be a variant of the refinement that is not as highly conservative, but rather adaptively performs refinement only where necessary to preserve bijectivity, as was done for the 2D case [Campen et al. 2016, Appendix B.2], though this is less obvious. Finally, our implementation used for evaluation is single-threaded. There are multiple opportunities for easy parallelization, such as growing stars in parallel, computing foliations in parallel, or performing the projections, arrangement computations, and tetrahedral subdivisions during refinement in parallel. There also is room to employ more efficient arithmetics, e.g. fast filtered predicates.

8 CONCLUSION

We have presented GalaxyMaps, a method to apply foliation based bijective map constructions in a local manner, reliably yielding global 3D bijective maps onto arbitrary convex or star-shaped domains. The local regions, stars, are adaptively determined as star-shaped regions around defects of imperfect non-bijective initial maps. Per star, our method makes use of a generalization of a previous foliation based mapping algorithm to more general domains.

In future work, besides addressing the items discussed in Section 7.4, it will be interesting to consider the combination of two (globally or locally) foliation based maps via an intermediate domain, so as to define bijective maps between two fully arbitrary shapes, similar to some approaches in the 2D setting [Schmidt et al. 2019; Weber and Zorin 2014]. This would enable reusing the reliable method presented herein, or parts thereof, for this even more general setting. Main challenges will lie in efficiently handling the task of piecewise linearization, amplified due to the composition of two maps.

ACKNOWLEDGMENTS

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

REFERENCES

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), 106:1–106:14.

Marc Alexa. 2023. Tutte Embeddings of Tetrahedral Meshes. *Discrete & Computational Geometry* (2023).

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-Rigid-As-Possible Shape Interpolation. In *Proc. SIGGRAPH 2000*. 157–164.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Hendrik Brückler, David Bommes, and Marcel Campen. 2022a. Volume Parametrization Quantization for Hexahedral Meshing. *ACM Trans. Graph.* 41, 4 (2022), 60:1–60:19.

Hendrik Brückler, Ojaswi Gupta, Manish Mandad, and Marcel Campen. 2022b. The 3D Motorcycle Complex for Structured Volume Decomposition. *Computer Graphics Forum* 41, 2 (2022), 221–235.

Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021. Efficient and Robust Discrete Conformal Equivalence with Boundary. *ACM Trans. Graph.* 40, 6 (2021), 261:1–261:16.

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

David Cohen and Mirela Ben-Chen. 2019. Generalized volumetric foliation from inverted viscous flow. *Computers & Graphics* 82 (2019), 152–162.

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.

Xingyi Du, Danny M. Kaufman, Qingnan Zhou, Shahar Z. Kovalsky, Yajie Yan, Noam Aigerman, and Tao Ju. 2022. Isometric Energies for Recovering Injectivity in Constrained Mapping. In *Proc. SIGGRAPH Asia 2022*. 36:1–36:9.

J. M. Escobar, E. Rodrıguez, R. Montenegro, G. Montero, and J. M. González-Yuste. 2003. Simultaneous untangling and smoothing of tetrahedral meshes. *Comput. Methods Appl. Mech. Eng.* 192, 25 (2003), 2775–2787.

Yu Fang, Minchen Li, Chenfanfu Jiang, and Danny M. Kaufman. 2021. Guaranteed Globally Injective 3D Deformation Processing. *ACM Trans. Graph.* 40, 4 (2021), 75:1–75:13.

Michael S. Floater. 1997. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design* 14, 3 (1997), 231–250.

Michael S. Floater and Valérie Pham-Trong. 2006. Convex combination maps over triangulations, tilings, and tetrahedral meshes. *Advances in Computational Mathematics* 25, 4 (2006), 347–356.

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.

Xiao-Ming Fu, Jian-Ping Su, Zheng-Yu Zhao, Qing Fang, Chunyang Ye, and Ligang Liu. 2021. Inversion-free geometric mapping construction: A survey. *Computational Visual Media* 7, 3 (2021), 289–318.

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

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.

Mark Gillespie, Boris Springborn, and Keenan Crane. 2021. Discrete Conformal Equivalence of Polyhedral Surfaces. *ACM Trans. Graph.* 40, 4 (2021), 103:1–103:20.

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.

Lisa Huynh and Yotam Gingold. 2015. Bijective Deformations in $\mathbb{R}^n$ via Integral Curve Coordinates. arXiv:1505.00073

S. S. Iyengar, Xin Li, Huanhuan Xu, Supratik Mukhopadhyay, N. Balakrishnan, Amit Sawant, and Puneeth Iyengar. 2012. Toward More Precise Radiotherapy Treatment of Lung Tumors. *Computer* 45, 1 (2012), 59–65.

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph.* 36, 6 (2017), 186:1–186:9.

Bert Jüttler, Sofia Maroscheck, Myung-Soo Kim, and Q Youn Hong. 2019. Arc fibrations of planar domains. *Computer Aided Geometric Design* 71 (2019), 105–118.

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.

Xin Li, Xiaohu Guo, Hongyu Wang, Ying He, Xianfeng Gu, and Hong Qin. 2007. Harmonic Volumetric Mapping for Solid Modeling Applications. In *Proc. SPM 2007*. 109–120.

Juncong Lin, Jiazhi Xia, Xing Gao, Minghong Liao, Ying He, and Xianfeng Gu. 2015. Interior structure transfer via harmonic 1-forms. *Multimedia Tools and Applications* 74, 1 (2015), 139–158.

Manish Mandad, Ruizhi Chen, David Bommes, and Marcel Campen. 2022. Intrinsic mixed-integer polycubes for hexahedral meshing. *Computer Aided Geometric Design* 94 (2022), 102078.

Tobias Martin, Guoning Chen, Suraj Musuvathy, Elaine Cohen, and Charles Hansen. 2012. Generalized Swept Mid-structure for Polygonal Models. *Computer Graphics Forum* 31, 4 (2012), 805–814.
Tobias Martin, Elaine Cohen, and Robert M. Kirby. 2008. Volumetric Parameterization and Trivariate B-spline Fitting using Harmonic Functions. In *Proc. SPM 2008.* 269–280.
I. Moerdijk and J. Mrčun. 2003. *Introduction to Foliations and Lie Groupoids.* Cambridge University Press.
Alexander Naitsat, Yufeng Zhu, and Yehoshua Y. Zeevi. 2020. Adaptive Block Coordinate Descent for Distortion Optimization. *Computer Graphics Forum* 39, 6 (2020), 360–376.
M. Nieser, U. Reitebuch, and K. Polthier. 2011. CubeCover – Parameterization of 3D Volumes. *Computer Graphics Forum* 30, 5 (2011), 1397–1406.
Valentin Z. Nigolian, Marcel Campen, and David Bommes. 2023. Expansion Cones: A Progressive Volumetric Mapping Framework. *ACM Trans. Graph.* 42, 4 (2023).
Matthew Overby, Danny Kaufman, and Rahul Narain. 2021. Globally Injective Geometry Optimization with Non-Injective Steps. *Computer Graphics Forum* 40, 5 (2021), 111–123.
Nico Pietroni, Marcel Campen, Alla Sheffer, Gianmarco Cherchi, David Bommes, Xifeng Gao, Riccardo Scateni, Franck Ledoux, Jean Remacle, and Marco Livesu. 2022. Hex-Mesh Generation and Processing: A Survey. *ACM Trans. Graph.* 42, 2 (2022), 16:1–16:44.
Roman Poya, Rogelio Ortigosa, and Theodore Kim. 2023. Geometric Optimisation Via Spectral Shifting. *ACM Trans. Graph.* 42, 3 (2023), 29:1–29:15.
Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 4 (2017), 16:1–16:16.
Patrick Schmidt, Janis Born, Marcel Campen, and Leif Kobbelt. 2019. Distortion-Minimizing Injective Maps Between Surfaces. *ACM Trans. Graph.* 38, 6 (2019), 156:1–156:15.
Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013), 125–135.
Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2 (2015), 11:1–11:36.
Breannan Smith, Fernando De Goes, and Theodore Kim. 2019. Analytic Eigensystems for Isotropic Distortion Energies. *ACM Trans. Graph.* 38, 1 (2019), 3:1–3:15.
Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4 (2015), 70:1–70:9.
Tommaso Sorgente, Silvia Biasotti, and Michela Spagnuolo. 2022. Polyhedron kernel computation using a geometric approach. *Computers & Graphics* 105 (2022), 94–104.
Jian-Ping Su, Xiao-Ming Fu, and Ligang Liu. 2019. Practical Foldover-Free Volumetric Mapping Construction. *Computer Graphics Forum* 38, 7 (2019), 287–297.
Jian-Ping Su, Chunyang Ye, Ligang Liu, and Xiao-Ming Fu. 2020. Efficient Bijective Parameterizations. *ACM Trans. Graph.* 39, 4 (2020), 111:1–111:8.
Thomas Toulorge, Christophe Geuzaine, Jean-François Remacle, and Jonathan Lambrechts. 2013. Robust untangling of curvilinear meshes. *J. Comput. Phys.* 254 (2013), 8–26.
Sofia Trautner, Bert Jüttler, and Myung-Soo Kim. 2021. Representing planar domains by polar parameterizations with parabolic parameter lines. *Computer Aided Geometric Design* 85 (2021), 101966.
W. T. Tutte. 1963. How to draw a graph. *Proc. Lond. Math. Soc.* 13 (1963), 743–767.
Yalin Wang, Xianfeng Gu, Tony F. Chan, Paul M. Thompson, and Shing-Tung Yau. 2004. Volumetric harmonic brain mapping. In *Proc. ISBI 2004.* 1275–1278.
Ofir Weber and Denis Zorin. 2014. Locally Injective Parametrization with Arbitrary Fixed Boundaries. *ACM Trans. Graph.* 33, 4 (2014), 75:1–75:12.
Jiazhi Xia, Ying He, Shuchu Han, Chi-Wing Fu, Feng Luo, and Xianfeng Gu. 2010. Parameterization of Star-Shaped Volumes Using Green's Functions. In *Proc. GMP 2010.* 219–235.
Yongjie Zhang, Wenyan Wang, and Thomas J. R. Hughes. 2012. Solid T-spline construction from boundary representations for genus-zero geometry. *Comput. Methods Appl. Mech. Eng.* 249–252 (2012), 185–197.
Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. arXiv:1605.04797

A MESH REFINEMENT

To (conservatively) generate a refined mesh M' on which the piecewise linearization of the foliation based map Ψ (as of Section 5) remains bijective, the edges of M are extruded through the mesh along the foliation, splitting the tetrahedra encountered. Per tetrahedron, the extrusion of an edge is planar due to the piecewise constant nature of the foliation's direction field d. The resulting blocks therefore are truncated polygonal prisms. These are then split into tetrahedra, pyramids, and triangular prisms by triangulating their polygonal bases. Pyramids and triangular prisms can then further be split into two and three tetrahedra, respectively. This yields the tetrahedral mesh M'. This mesh has the property that two distinct leaves of the foliation that pass through a common tetrahedron, pass only through common tetrahedra all along. Hence, they pass through the same stack of tetrahedra, from center to boundary. Based on this property, bijectivity of the piecewise linearized map Ψ' on M' is easily shown [Campen et al. 2016, Prop. 8].

Implementation. The above conceptual procedure can be implemented as follows. Rather than incrementally performing splits, edge by edge and tetrahedron by tetrahedron, our proposed implementation first plans the complete refinement of M, and then constructs the refined mesh M'. This avoids intermediate non-conforming configurations (refined elements adjacent to not yet refined ones) that are challenging to handle in the three-dimensional polyhedral setting and require complex data structures. The refinement plan is stored as 2D *patterns* on the triangles of M. These patterns describe the bases of the above mentioned truncated polygonal prisms, that will be divided into the tetrahedron stacks. First, every edge of M is iteratively projected through the tetrahedra (along the foliation direction) inwards until p_0 and outwards until the boundary ∂M is reached. On every triangle that is hit in this process, the edge's projection (a line segment) is stored in this triangle's pattern. Each such pattern is represented as a 2D *arrangement* (we use the implementation from CGAL). In the end, the pattern on each triangle of M is a polygonal partition of the triangle, induced by the straight line segments. Next, we triangulate these pattern polygons in a consistent manner: A polygon of one triangle is triangulated, and the introduced additional line segments again projected through the mesh along the foliation. This triangulates all polygons of the same stack (pierced by the same bundle of leaves) in the same way. For each tetrahedron, the (now triangular) patterns on its four face triangles define a set of triangular truncated prisms (consisting of triangular prisms, pyramids, and tetrahedra) partitioning the tetrahedron. Hence, we can now incrementally build M' from scratch: For each prism, implied by the patterns on M, a set of (commonly 1–3) tetrahedra (partitioning this prism) is added to M'. Special care must be taken to ensure conformance of these tetrahedra: The truncated prisms share triangular or quadrangular faces. The same diagonal of quadrangular faces must be used when defining the tetrahedra for the two adjacent prisms, i.e. these diagonals need to be chosen and fixed per quadrangular face in advance. Note that a prism whose 3 quadrilateral faces have diagonals fixed in the same direction cannot be filled with just 3 tetrahedra. This case can either be avoided (by computing a globally consistent orientation for the diagonals) or such a prism simply be filled with 8 tetrahedra around an additional vertex, positioned in the center of its 6 neighbors in object space as well as in image space.

Simplification. To reduce the amount of refinement we employ the strategies described in [Campen et al. 2016, Appendices B.1 and B.3]: Before refinement, the foliation directions are greedily adjusted to align to edges and faces, reducing the number of stacks implied. After refinement, the subtetrahedralizations per original tetrahedron are decimated as long as bijectivity is maintained.