Simplicial Complex Augmentation Framework for Bijective Maps

ONLINE SUBMISSION ID: 0375

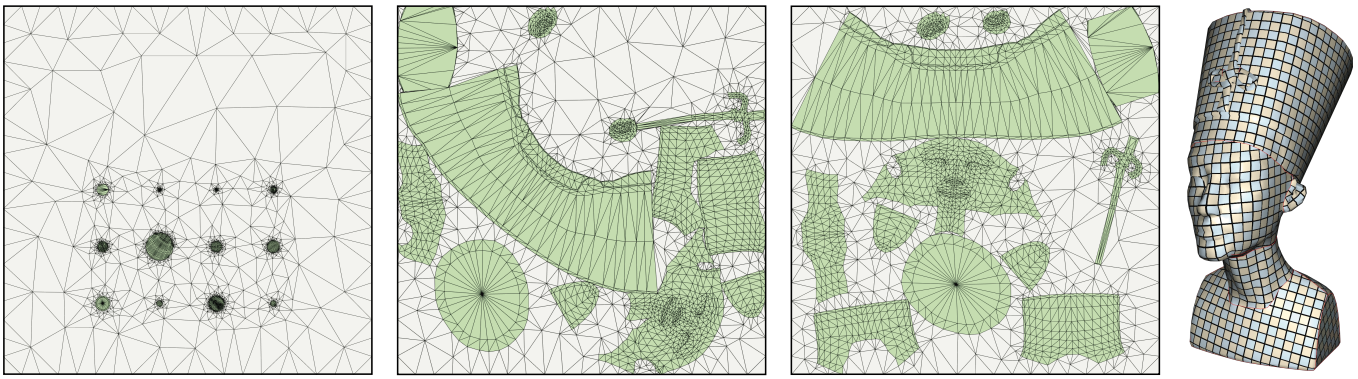

Fig. 1. The Nefertiti model, with prescribed seams, is UV mapped by our algorithm. From left to right: each UV chart is bijective mapped into a circle using Tutte's embedding, the distortion is minimized for all charts. The layout is further improved using interactive tools and the final parametrized model is shown on the right. The automatic parametrization took less than a second, and our approach is guaranteed to produce a valid UV map with no inverted elements or self-overlapping triangles. We attach a video of the optimization and manual interaction in the additional material.

Bijective maps are commonly used in many computer graphics and scientific computing applications, including texture, displacement, and bump mapping. However, their computation is numerically challenging due to the global nature of the problem, which makes it prohibitively expensive to optimize with standard smooth optimization techniques.

We propose to use a scaffold structure to reduce this challenging and global problem to a local injectivity condition. This construction allows us to benefit from the recent advancements in locally injective maps optimization to efficiently compute large scale bijective maps (both in 2D and 3D), sidestepping the need to explicitly detect and avoid collisions.

Our algorithm is guaranteed to robustly compute a globally bijective map, both in 2D and 3D. To demonstrate the practical applicability of our method, we use it to compute globally bijective single patch parametrizations, to pack multiple charts into a single UV domain, to remove self-intersections from existing models, and to deform 3D objects while preventing self-intersections.

Our approach is simple to implement, efficient (two orders of magnitude faster than competing methods), and robust, as we demonstrate in a stress test over a parametrization dataset with 103 models.

ACM Reference format:
Online Submission ID: 0375. 2017. Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph.* XX, X, Article XXX (November 2017), 8 pages.
https://doi.org/0000001.0000001_2

1 INTRODUCTION

The computation of discrete maps is a fundamental problem in computer graphics that has been extensively studied in the last three decades. The problem is challenging due to the large solution space, and the non-linearity of the desired properties (both in the distortion measures and in the desired constraints). Algorithms for robustly and efficiently computing locally injective (i.e. non-flipping) maps have been only recently introduced [Lipman 2012; Kovalsky et al. 2016; Rabinovich et al. 2017] and are now having a major impact in many research areas outside of traditional texture mapping, including remeshing [Bommes et al. 2013], image editing [Poranne and Lipman 2014], and cultural heritage [Pal et al. 2014].

In this paper, we consider the problem of generating bijective maps, i.e. locally injective maps that do not have self-overlapping triangles. This is a difficult problem, exacerbated by the fact that any pair of triangles could potentially overlap, leading to a quadratic number of non-linear constraints to satisfy. This problem is usually tackled by iteratively deforming an existing map, checking for overlaps after each step, and then preventing the overlap using constraints [Harmon et al. 2011] or penalty forces [Harmon 2010]. These methods require a spatial acceleration structure to find the candidate pairs of overlapping elements and a resolution strategy that updates the map while avoiding the detected overlaps. The newly computed displacement might lead to new overlaps, and this process is thus performed iteratively until no new collisions are found. Difficult cases with many collisions might requires tens or hundreds of iterations before all the candidate intersecting pairs are detected.

Our approach sidesteps the need to find candidate self-intersections using a simple observation: if the the entire ambient space is tessellated and the boundary is fixed, then local injectivity implies global bijectivity [Zhang et al. 2005; Lipman 2013; Müller et al. 2015]. We propose an optimization framework based on this idea that leverages recent techniques for injective maps; our algorithm is simple to implement, robust, and two orders of magnitude faster than competing methods.

Overview. Given an input bijective map represented by a discrete triangle mesh and its mapped vertex locations, we create a new scaffold mesh for a bounding box that contains the initial, mapped triangle mesh (Figure 1) and conforms to it's boundary. We then

2017. 0730-0301/2017/11-ARTXXX $15.00
https://doi.org/0000001.0000001_2

optimize for the desired property of the map (such as distortion, positional constraints, etc.) while ensuring that no triangle will flip. This property is achieved using a variational formulation that combines a user-defined energy for the map with a regularization term that allows the scaffold to freely deform without hindering the optimization of the map properties. During the optimization, we refine and optimize the connectivity of the scaffold mesh to prevent possible locking situations.

We demonstrate the practical utility of our algorithm in the context of single patch mesh parametrization by producing distortion minimizing bijective maps for a collection of 103 challenging models. Our algorithm is ideal to compute tight UV maps for models with multiple connected components and seams as we demonstrate in our interactive texture packing experiments. Our algorithm can also be easily extended to 3D by replacing the triangular scaffold with one composed of tetrahedra. For the 3D case, we show that our method can be used to deform surfaces preventing self-intersections and to remove self-intersection from existing genus-0 surfaces when paired with a mean conformalized flow [Kazhdan et al. 2012; Sacht et al. 2013].

In the additional material, we provide a video (showing the optimization iterations) and the input/output meshes for each figure in the paper. To foster replicability of results, we will release an open-source reference implementation of our algorithm.

2 PREVIOUS WORK

Bijective maps find a host of applications in a variety of fields including physical simulation, surface deformation, and parametrization. We review only the most relevant prior works here and refer to the following surveys for more details [Floater and Hormann 2005; Sheffer et al. 2006; Hormann et al. 2007].

Locally Injective Maps.

There are many methods that focus on creating injective maps, which amounts to requiring that triangles maintain their orientation (i.e. they do not flip). In mesh parameterization, many flip-preventing metrics have been developed: the idea is to force the metric to diverge to infinity as triangles become degenerate, inhibiting flips. These metrics optimize various geometric properties such as angle [Hormann and Greiner 2000; Degener et al. 2003] or length [Sander et al. 2001; Sorkine et al. 2002; Aigerman et al. 2014; Poranne and Lipman 2014; Smith and Schaefer 2015] preservation. Similar techniques in the context of deformation have been used to add barrier functions to enforce local injectivity in deformations [Schüller et al. 2013]. Our method uses these techniques to prevent flips in the scaffold.

Many methods have also been developed to optimize these distortion energies including moving one vertex at a time [Hormann and Greiner 2000], parallel gradient descent [Fu et al. 2015], as well as other quasi-newton approaches [Smith and Schaefer 2015; Kovalsky et al. 2016; Rabinovich et al. 2017]. Other approaches construct such maps by performing a change of basis, projecting to an inversion free space, and then constructing a parametrization from the result [Fu and Liu 2016]. While our method could potentially use any of these optimization methods, we use [Rabinovich et al. 2017] for its large step sizes. We elaborate on this choice in Section 3.1.

Bijective Maps.

In addition to injective constraints, bijective maps have the additional requirement that the boundary does not intersect. One simple method for creating a bijective map in 2D involves constraining the boundary to a convex shape such as a circle [Tutte 1963; Floater 1997]. Such parametrizations guarantee a bijective map in 2D but create significant distortion. Even so, these methods are commonly used to create a valid starting point for further optimization [Schüller et al. 2013; Smith and Schaefer 2015; Rabinovich et al. 2017]. While methods that produce bijective maps with fixed boundaries exist [Weber and Zorin 2014; Campen et al. 2016], we aim to produce maps where the boundary is free to move to reduce the distortion of the map.

[Gotsman and Surazhsky 2001; Zhang et al. 2005] provide a method where the free space is triangulated with a scaffold. The scaffold triangles are given a step function for their error that is zero unless a triangle flips where the error is infinity. Hence, the bijective condition becomes local in that the shape can evolve until a scaffold triangle flips, in which case the free space is retriangulated and the optimization continues. The main limitation of this work is the lack of an evolving triangulation during the line search and the absence of a rotationally invariant metric for the scaffold triangles, which lead to very small steps and an inefficient optimization.

The Deformable Simplicial Complex (DSC) method [Misztal and Bærentzen 2012] utilize a triangulation of both the free space and the interior of an object to track the interface between the two volumes. Similar to [Zhang et al. 2005], the DSC retriangulates at degeneracies but also performs operations to improve the shape of the triangles. This method changes the triangulation of the interface that it tracks, which works well for simulation, but it is not allowed in many other applications such as UV mapping.

Air meshes [Müller et al. 2015] extends the technique of Zhang et al. [2005] to add the concept of triangle flipping based on a quality measure during the optimization instead of simply retriangulating at the first sign of a degeneracy. However, this method does not maintain bijective maps as boundaries are allowed to inter-penetrate during optimization: the scaffold is only used to efficiently detect problematic regions, and the local injectivity requirement is a soft constraint in the optimization. The problem tackled in this paper is much harder, because we do not allow any overlap during any stage of the optimization to guarantee that the resulting maps will be bijective.

Smith et al. [Smith and Schaefer 2015] take a different approach: instead of using a scaffold triangulation, the authors introduce a locally supported barrier function for the boundary to prevent intersection and explicitly limit the line search by computing the singularities of both the distortion energy and the boundary barrier function. Such an approach is inspired by traditional collision detection and response methods that are discussed below. Given a bijective starting point, this approach never leaves the space of bijective maps during optimization. Its main limitation is that it is computationally expensive, especially for large models. Our method is two order of magnitude faster (Figure 11).

Collision Detection and Response.

While not directly related to our approach, bijective maps inherently involve some form of collision detection and response to avoid overlaps. The field on collision detection is vast, and we refer the reader to a survey [Jimenez et al. 2001]. In terms of simulations, methods such as asynchronous contact mechanics [Harmon et al. 2009; Harmon 2010; Ainsley et al. 2012] ensure the bijective property but are very expensive and designed to operate as part of a simulation. Differently, our approach is specialized for geometric optimization, where we are interested in a quasi-static solution (i.e. we do not want to explicitly simulate a dynamic system, but only find an equilibrium solution).

The work that is closer to ours in term of application (but very different in term of formulation) is [Harmon et al. 2011], where collision detection and response is used to interactively deform shapes while avoiding self-intersections. Similarly to the previous methods, the explicit detection and iterative response is expensive when many collisions happen at the same time. Our work avoids these expensive computations, and can robustly handle hundreds of simultaneous collisions while still making large steps in the optimization.

Seam Creation.

In the context of parametrization, some approaches optimize the connectivity of the charts of the surface during parametrization to obtain a bijective map. [Lévy et al. 2002; Zhou et al. 2004] parameterize the surface and then split charts based on whether they intersect [Lévy et al. 2002] or based on a level of distortion [Zhou et al. 2004]. Sorkine et al. [Sorkine et al. 2002] employ a bottom-up approach and add triangles to a parametrization chart until bijectivity would be violated. The problem we are solving is more general (seams are only useful for texture mapping applications) and constrained (we preserve the prescribed seams). Our algorithm could be used by these algorithms to parametrize single charts, which could reduce the number of additional seams.

3 METHOD

Our method, Simplicial Complex Augmentation Framework (SCAF) utilizes a scaffold structure to robustly compute a bijective map between a pair of simplicial meshes with the same connectivity. SCAF is specialized for the context of geometric optimization, i.e. we are interested in maps with low distortion and optionally satisfying a set of geometric constraints. We assume our maps are continuous and piecewise affine, i.e. the map deforms every simplex with an affine deformation. Thus, we can fully define the map using the image of its vertices.

Our algorithm uses a discrete, bijective identity map (encoded as a non-overlapping and non-flipping triangle/tetrahedral mesh) as initialization and then iteratively refines it, displacing the vertices while always ensuring that it remains bijective. Our method is composed of three stages: (1) we augment the initial mesh with a scaffold, filling a bounding box around the initial map image; (2) we optimize both the extended mesh (scaffold included), reducing the geometric distortion of the map; (3) we update the vertices and scaffold, enlarging the bounding box if necessary, to improve the quality of the triangulation. Steps (2) and (3) are iterated until the quality of the map is deemed sufficient.

3.1 General Formulation

Denote the input simplicial mesh by $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ with a single, simple boundary representing a compact d-dimensional manifold embedded in the d-dimensional Euclidean space, where $\mathcal{V}$ is the set of n-vertices and $\mathcal{F}$ is the set of m-simplices. Our goal is to compute a continuous and piecewise affine mapping $\Phi : \mathcal{M} \to \mathbb{R}^d$ with $\Omega := \Phi(\mathcal{M}) = (\mathcal{V}', \mathcal{F})$ the resulting simplicial mesh with the same connectivity as $\mathcal{M}$.

We are interested in the bijective map that minimizes a given type of geometric energy:

$$\begin{aligned} \min_{\mathcal{V}'} \quad & E_{\mathcal{M}}(\Phi) \\ \text{s.t.} \quad & \Phi \text{ is bijective,} \end{aligned} \tag{1}$$

where $E_{\mathcal{M}}$ is a user-defined geometric energy.

Reduction to Local Orientation Preservation. A sufficient condition for the simplicial map $\Phi : M \to \Omega$ to be bijective is that the map preserves orientation and its restriction to the boundary $\Phi|_{\partial M} : \partial M \to \partial\Omega$ is bijective [Lipman 2013]. In this light, we are able to take advantage of the following simple construction (Figure 2): the algorithm extends the axis aligned bounding box of M to get a d-orthotope and fills the enclosed region with a scaffold simplicial complex to form a simplicial complex mesh D that includes and conforms to $M \subset D$. We can now define the continuous and piecewise affine map $\Psi : D \to D'$ with $\Phi = \Psi|_M$ where $\Psi|_{\partial D}$ is the identity map, and denote the scaffold region as $S = D \setminus M$. Now Φ is guaranteed to be bijective if Ψ preserves orientation. Using this observation, we can translate the bijectivity into a local, orientation-preserving requirement defined per simplex.

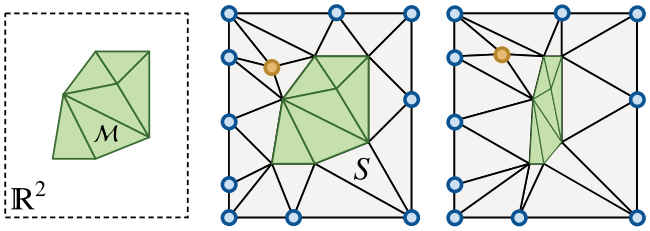

Fig. 2. The initial mesh $\mathcal{M}$ (in green, left), is embedded in another mesh D (in gray, middle) that covers a box in the ambient space and contains the same triangles as $\mathcal{M}$. D might contain additional points (orange). We denote the triangles that are in D but not in $\mathcal{M}$ as the scaffold S. Our algorithm deforms D, inducing a corresponding deformation on $\mathcal{M}$ (right), while keeping the boundary (blue vertices) fixed and preventing changes in the triangle orientation.

Variational Formulation. Minimizing the distortion of Φ using the augmented map Ψ poses an interesting challenge: what is the desired shape of the scaffold S? Ideally we would like the simplices in the scaffold to maintain their orientation and not affect the optimization in any other way. Such a requirement is difficult to model directly, since it is a discontinuous condition that is not well-suited for the variational framework that we would like to use to minimize $E_{\mathcal{M}}$.

We propose a regularized version of this condition modeled with an energy $E_{\mathcal{M}}(\Psi|_S)$ that still diverges when elements change orientation and that mildly penalizes any non-rigid distortion.

We choose a reweighted version of the symmetric Dirichlet energy $\mathcal{D}$ [Smith and Schaefer 2015], measured w.r.t. the Jacobian of the map Ψ computed from the rest pose of S for each simplex f,

$$J_f := \nabla \Psi_f \tag{2}$$

where Ψ_f is the restriction of Ψ over the simplex f, which is an affine map. We divide the energy of each scaffold simplex f by its area A_f, and sum them up to obtain the final energy that favors an equal contribution regardless of the size of each scaffold simplex:

$$\begin{aligned} E_S(\Psi|_S) &= \sum_{f \in S} \frac{1}{A_f} \mathcal{D}(J_f) \\ &= \sum_{f \in S} (\|J_f\|_F^2 + \|J_f^{-1}\|_F^2 - 2d). \end{aligned} \tag{3}$$

The $-2d$ term ensures that the energy is 0 when $J_f = \mathbb{I}$. The map is then computed by summing the two terms:

$$\begin{aligned} \min_D \quad & E_{\mathcal{M}}(\Psi|_{\mathcal{M}}) + \lambda E_S(\Psi|_S) \\ \text{s.t.} \quad & \Psi|_{\partial D} = \mathbb{I} \\ & \Psi \text{ preserves orientation.} \end{aligned} \tag{4}$$

where $\lambda > 0$ is balancing the contribution of the two energies, decreasing as the optimization proceeds.

Iterative Regularization. Solving this problem leads to a bijective and distortion minimizing map, but the regularizer will affect the stationary points of $E_{\mathcal{M}}$, which is problematic, especially for large deformations. To address this problem we iteratively minimize this energy, regenerating the scaffold at each iteration, and use the new scaffold as a rest pose for the regularization term $E_S(\Psi|_S)$. This iterative procedure has two positive effects: (1) it acts as a proximal regularization term without inhibiting movement since the rest pose is updated at each iteration; (2) the meshing quality of the scaffold is high, which avoids locking configurations.

Interpolation Coefficient. We experimentally observed that our algorithm is robust to different choices of λ, generating indistinguishable results in most cases. However, λ affects the convergence speed (Figure 3). We used $\lambda = \frac{1}{100} \frac{E_{\mathcal{M}}(\Psi|_{\mathcal{M}})}{|S|}$ for all our experiments, where $|S|$ is the number of scaffold simplices.

Solver. Since our energy is rotational invariant, we can minimize the energy with the same quadratic proxy proposed in SLIM [Rabinovich et al. 2017], enriching the approach with the equality constraint needed to fix the boundary ∂D. We also employ the orientation-preserving line search [Smith and Schaefer 2015] (with exact predicates [Shewchuk 1996]) to ensure that no triangles can change orientation. Alternatively, other methods such as AQP [Kovalsky et al. 2016] could be used to minimize this energy. Since our approach changes the mesh connectivity at every iteration, AQP loses much of its advantages as the approximate hessian must be recomputed at each iteration and not prefactored. Therefore, our approach is an ideal fit for [Rabinovich et al. 2017], which takes large steps at every iteration without relying on a constant, prefactored matrix at each iteration. For practical applications, SLIM iterations are sufficient to minimize the energy to acceptable levels. For two stress tests (figures 4 and 11) we used the result of SLIM as a warm start for a Newton optimization (as suggested by [Rabinovich et al. 2017]), which quickly converges to a numerical minimum.

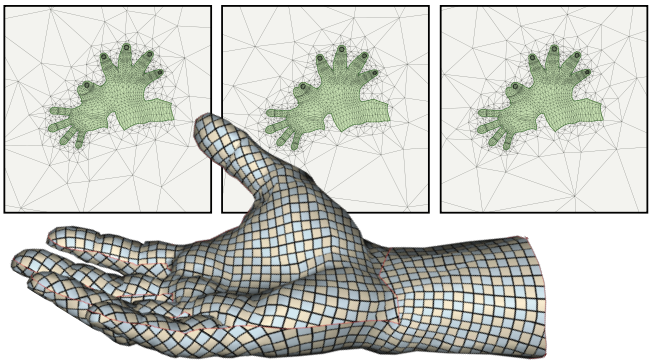

Fig. 3. Different values of λ do not affect the result, but they change the number of iterations needed. From left to right: we used a large weight (100x ours), our weight, and a small weight (0.01x ours). The optimization took 9,7, and 8 iterations, respectively, to reach the same energy level.

3.2 Surface Parametrization

Our framework can be used for many applications, one of which is computing a bijective surface parametrization from a 3D surface into the UV plane. We follow [Liu et al. 2008] and assume that each 3D triangle f^{3D} is equipped with a rigid transformation R_f such that applying R_f to f^{3D} maps f^{3D} to the plane. Given this transformed triangle f, we can now measure the distortion of the map using the Jacobian of the affine transformation (a 2×2 matrix) from $R_f(f^{3D})$ to f, the location of the triangle in the parametrization.

Initialization. Our method is initialized with Tutte's embedding algorithm [Tutte 1963]:

$$\Phi^0 : \mathcal{M}^{3D} \rightarrow \Omega^0,$$

where Ω^0 is a simplicial disk domain. Then we construct a larger rectangular domain $D^0 \supset \Omega^0$, where ∂D^0 is an axis-aligned rectangle, and use Triangle [Shewchuk 1996] to triangulate the region in between. We enforce a quality bound of 20° to obtain a graded mesh that is coarse on the boundary and conforming the boundary of Ω^0. This grading implicitly produces an approximate inverse distance weighting of the scaffold space with respect to the error function, which enables a larger deformation per iteration. Then we define $\Psi : D^0 \rightarrow D \subset \mathbb{R}^2$ and restrict $\Psi|_{\partial D^0}$ to be the identity.

Mesh Improvement. At the end of each iteration, we improve the quality of the scaffold. The reason for maintaining a good mesh quality is two-fold. First, as observed in [Zhang et al. 2005; Müller et al. 2015], fixing the scaffold will potentially prevent movement. Secondly, the quality of the scaffold affects the condition of the linear system in SLIM [Rabinovich et al. 2017]: a higher quality leads to larger and more efficient iterations.

We resort to Triangle [Shewchuk 1996] to create the initial scaffold and to regenerate the scaffold mesh in the improvement step. Our experiments show that, in 2D, it is faster to generate the mesh from scratch at every iteration instead of trying to optimize the scaffold using local operations as suggested in [Müller et al. 2015]. Since our solver makes large steps in each iteration, the scaffold requires significant connectivity changes each iteration, which explains why regenerating the triangulation is faster than local operations. This is in stark contrast with physical simulation scenarios where each iteration represents a small time step and, thus, a minor change in the vertex positions.

We demonstrate the effectiveness of the remeshing strategy in Figure 4 where our method recovers from a large rotation — note that the scaffold is updated during the iterations and always leaves space for the map to move freely.

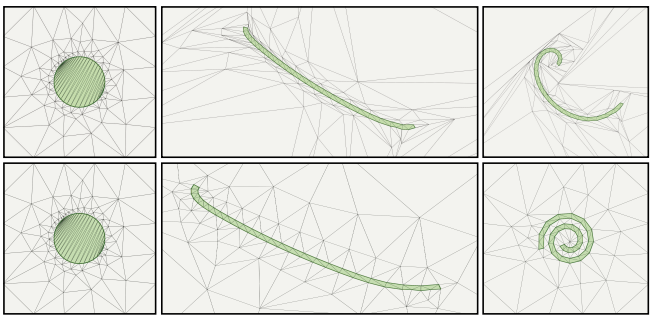

Fig. 4. A bijective map from a circle (left) to a spiral (right) is computed without (top) and with (bottom) the iterative remeshing step. The slivers in the triangulation locks the optimization (top), preventing it from reaching the target shape.

Sliding. As the optimization proceeds and some of the boundary elements get closer, some of the scaffold triangles might (and often will) get smaller and smaller, restricting the amount of sliding that is allowed in one iteration as well as introducing numerical difficulties in computing the corresponding Jacobian J_f.

To avoid this issue, we replace the target Jacobian for the triangles with an area smaller then ϵ with the transformation that maps the small triangle to an equilateral triangle with area ϵ. To set ϵ, we traverse through the boundary of the interior of the uv domain at the current iteration to find the minimum edge length l. In our implementation, $\epsilon = \frac{l^2}{4}$.

A theoretical downside of this modification it affects the distortion energy. We experimentally observed that the changes are negligible, and we thus used it for all our experiments. A practical upside is that it discourages fully degenerate elements — this change, coupled with the orientation preserving line search [Smith and Schaefer 2015], makes our algorithm extremely robust, even on challenging stress tests (Figure 5).

3.3 Extension to 3D

Our algorithm readily extends to the 3D case with only one major difference: the scaffold becomes a tetrahedral mesh, which is computationally more challenging to create and update.

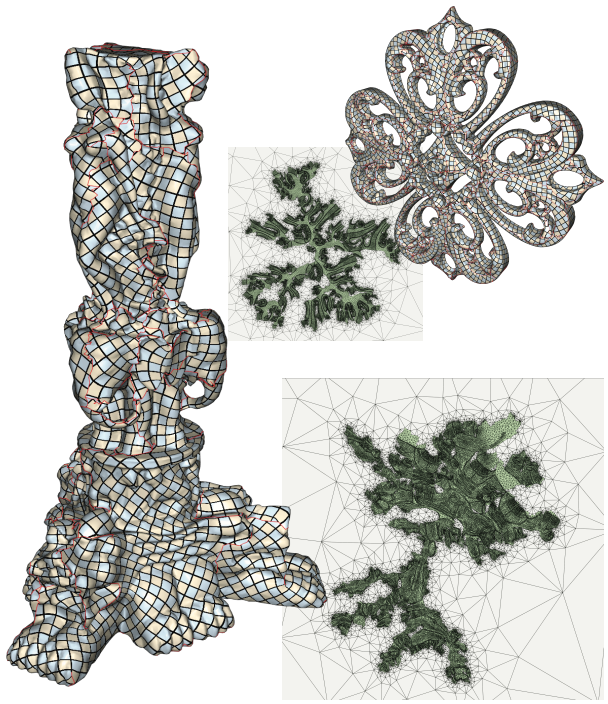

Fig. 5. Two models are cut using [Bommes et al. 2009] and bijectively parametrized using our algorithm. See the additional material for more examples.

We use Tetgen [Si 2015] to generate the initial scaffold and the local operations proposed in [Klingner 2009] to optimize the scaffold's quality in the subsequent iterations. It is unfortunately not possible to directly use Tetgen at every iteration as we did with Triangle in the 2D case, since Tetgen fails when boundaries get too close, which is common in our experiments.

4 RESULTS

We implemented our algorithm in C++ using Eigen for linear algebra routines. We run our experiments on a desktop with a 4-core Intel i7 processor clocked at 4 Ghz and 32 Gb of memory using only one thread. For all experiments, the scaffold bounding box is computed by uniformly scaling by three times the bounding box of the image of the current map.

Robustness. To demonstrate the robustness of our algorithm, we computed bijective maps for all the 103 meshes parametrized by the MIQ algorithm [Bommes et al. 2009] in the dataset proposed by [Myles et al. 2014]. The cuts in these meshes have been designed for locally injective parametrization that usually have major self-overlap. We use them as a stress test for the effectiveness and robustness of our method: the cuts introduce a massive distortion in the Tutte initialization and lead to boundaries that are prone to overlap in hundreds of locations. Our method successfully creates bijective parametrizations for all these models with default parameters. We attach all the parametrized models in the additional material and show two examples in Figure 5.

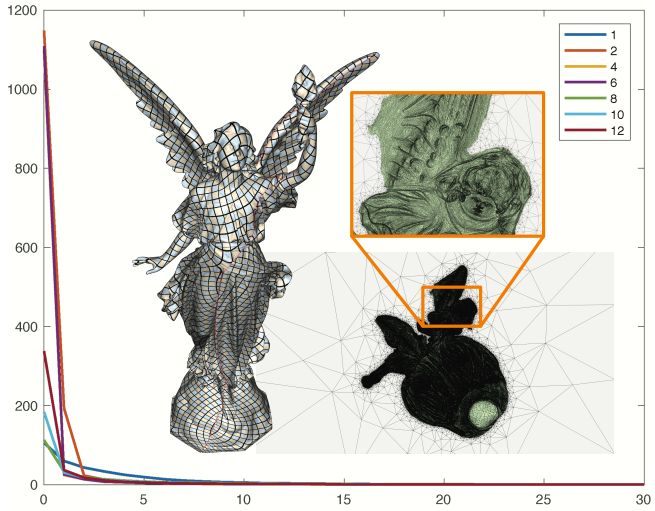

Fig. 6. We compare the distortion energy with respect to the number of iterations on a set of Lucy's meshes with different resolutions (from 1 to 12 million faces). In the center of the plot, we show the 1M Lucy model parametrized by our algorithm.

Scalability. Our methods scales gracefully to large datasets, similarly to [Rabinovich et al. 2017]. We repeat their scalability experiment, but producing bijective maps instead of just locally injective maps (Figure 6). The behaviour is remarkably similar — the density of the model (and consequently of the scaffold) does not affect the number of required iterations.

Texture Atlas Generation. UV mapping is a time consuming procedure required in most geometric modeling pipelines. Existing commercial tools provide the ability to flatten single patches and arrange them in UV layouts where multiple patches are tightly packed inside a rectangular domain, which is then loaded in the texture memory of a GPU.

Our algorithm can bijectively parameterize a single patch (Figure 7), avoiding the typical manual UV postprocessing required with traditional tools. Our algorithm can also be used to create automatic UV charts of models with multiple connected components (or predefined cut edges). We show an example in Figure 8 where we detected the connected components, bijectively map the patches into a set of circles (using a grid layout), and reduce their distortion using our algorithm. The result is a tight and automatic packing without resorting to any user-interaction. Additional interactive tools can further improve the atlas by dragging&dropping regions or translating islands while ensuring that no overlaps are introduced (Figure 1). We show interactive sessions using our packing tool in the additional material.

Preventing Self-Intersections. Our algorithm can be generalized to handle mixed dimension problems, such as the deformation of 2D surface in 3D space, while preventing self-intersections. In Figure 9, we demonstrate the use of our method to resolve self-intersections of surfaces. First we perform a conformalized flow [Kazhdan et al. 2012] using the algorithm proposed in [Sacht et al. 2013] to resolve

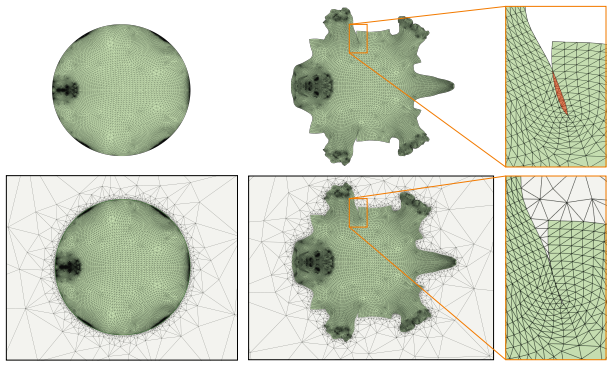

Fig. 7. A mesh is cut by an artist into a single chart and parametrized using SLIM [Rabinovich et al. 2017] (left) and with our algorithm (right). Note that local-injectivity is not sufficient for this model, since the global overlaps in the highlighted region prevent this parametrization from being as a UV texture map. Our result (right) is guaranteed to be bijective.

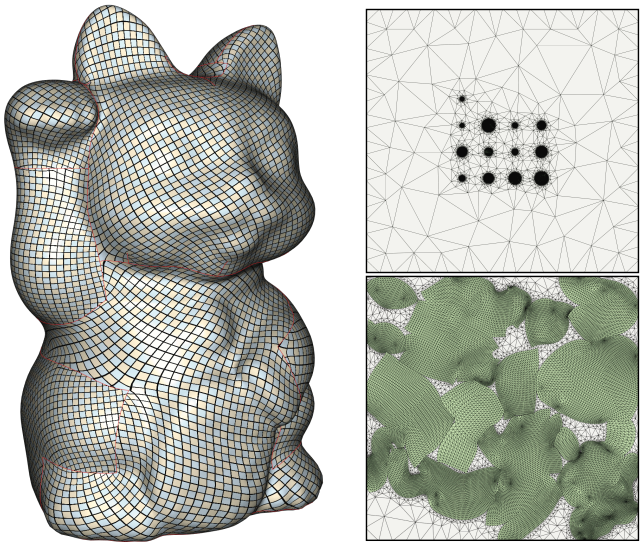

Fig. 8. A model with multiple chart (left) is automatically parametrized in a texture atlas (bottom-right) by first mapping each component to a circle (top-right) and then minimizing the distortion.

any self-intersections. While [Sacht et al. 2013] will resolve the intersections, the resulting surface may be geometrically far from the initial shape (see Figure 9). Next we tetrahedralize the ambient space while conforming to the deformed surface mesh and minimize Equation 4 with an additional energy term that strives to restore the rest pose geometry of the surface, using the surface ARAP energy proposed in [Sorkine and Alexa 2007]. The result is a surface similar to the original mesh, but without self-intersections. In this example, it is possible to observe that even dramatic changes of scale (on the foot) can be robustly handled by our parametrization algorithm.

A more challenging stress test is shown in Figure 10, where the bunny model is scaled up inside a box, to 30 times its original size. No

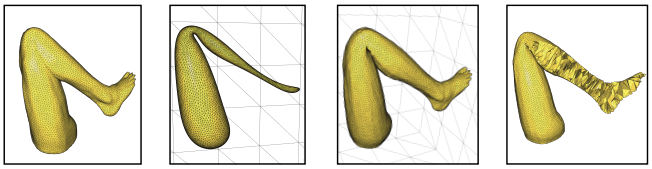

Fig. 9. We remove the self-intersections from a genus 0 model using the conformalized flow [Kazhdan et al. 2012; Sacht et al. 2013]. The flow is inverted, while using our algorithm to compute a bijective volumetric map, to recover a self-intersection free version of the original surface. The final model can now be meshed using tetgen, since it is free from self-intersections.

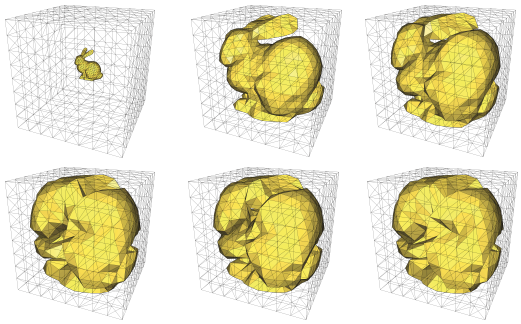

Fig. 10. We grow a bunny inside a box, while preventing self-intersections. We show the result after 0,10,20,30,40, and 50 iterations.

self-intersections are introduced, despite the extreme, constrained deformation.

Comparison with [Smith and Schaefer 2015]. The algorithm that is closest to ours is [Smith and Schaefer 2015], which tackles a similar problem (restricted to the 2D case). We replicated the space filling curve experiment and obtained remarkably similar results, where our running time is 96s, compared with 8472s for [Smith and Schaefer 2015] (88 times faster). We show in Figure 11 a more challenging experiment with a subdivided version of the space filling curve to emphasize the performance difference: our algorithm converges in 39 minutes, while [Smith and Schaefer 2015] did not converge after 5 days and 21 hours. For this example, we used the procedure suggested in [Rabinovich et al. 2017]: we performed a few iterations minimizing the quadratic proxy and then switch to a traditional newton method until numerical convergence. A video of the optimization is provided in the additional material.

Local vs Global Optimization. Both [Zhang et al. 2005] and [Misztal and Bærentzen 2012] use a construction similar to ours to generate bijective maps (Section 2). Both methods explicitly prevent changes of orientation using a local approach: they optimize the map using coordinate descent iterations [Solomon 2015] allowing only one vertex at a time to move in its 1-ring and thus ensuring that no triangle flip. This strategy severely limits the maximal displacement per iteration and restricts the step to the size of the 1-rings. Such a restriction makes these methods impractical for parametrization applications since the difference in scale between the Tutte's embedding and the final result is extreme (the ratio of min and max

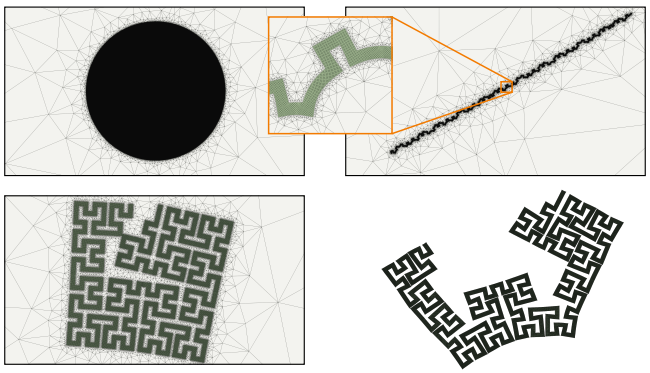

Fig. 11. [Smith and Schaefer 2015] and our algorithm produce visually similar results. However, our algorithm (right) is 200 times faster than [Smith and Schaefer 2015] (left), producing this result in only 39 minutes, compared to more than 5 days for [Smith and Schaefer 2015].

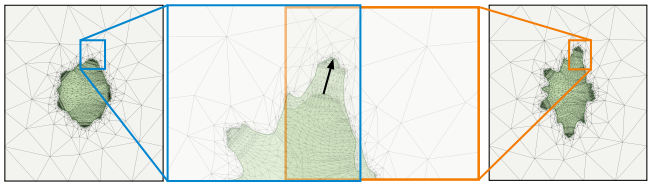

Fig. 12. A single iteration of our algorithm (from left to right) drastically reduces the distortion. The black vector in the center is 150 times longer than the average edge length of its 1-ring. Iterative methods would need thousands of iterations to achieve a similar progress.

Dataset	#V	#F	$\#V_S$	$\#F_S$	It.	Time (s)
Nefertiti (Fig. 1)	1704	2823	749/ 207	1998 /914	26	0.56
Hand (Fig. 3)	2239	4046	347/280	1104/970	7	0.14
Spiral (Fig. 4)	54	52	78/36	190/106	50(50)	0.04(0.21)
Thai Statue (Fig. 5, left)	42405	79970	3665/1593	12148/8004	50	28.28
Filigree (Fig. 5, right)	56062	100000	9160/2627	30422/17356	100	75.99
Lucy (Fig. 6)	501105	1000000	1856/ 3470	5900/ 5674	100	2524.22
Lucy (Fig. 6)	1001375	1999999	2284/ 4400	7297/ 7133	100	7251.00
Lucy (Fig. 6)	2002031	3999999	3587/ 6930	11215/ 10985	100	22500.07
Lucy (Fig. 6)	3002899	5999999	5135/ 9859	16047/ 15601	100	52235.31
Lucy (Fig. 6)	4002816	8000000	5140/ 10288	15890/ 15918	100	59413.14
Lucy (Fig. 6)	5003408	10000000	6194/ 12231	19182/ 19040	100	95247.59
Animal (Fig. 7)	19937	39040	747/593	2306/1998	50	15.36
Maneki-Neko (Fig. 8)	23025	43648	2427 /725	7174/ 3770	50	16.81
Leg (Fig. 9)	6617	13230	5016/5021	68521/68544	500	3251.17
Bunny (Fig. 10)	568	1132	683/706	6209/6289	50	7.16
Space Filling (Fig. 11)	79545	146832	90815/88237	181608/176452	200(250)	547.13(1836.58)
Camel (Fig. 12)	2032	3576	384/323	1234/1112	10	0.19

Table 1. Timings and statistics for the models shown in the paper. From left to right: number of input vertices and simplices, number of initial/final scaffold vertices and simplices, number of iterations, running time in seconds. The numbers in parenthesis refer to the Newton optimization. Note that our timings are considerably higher than those reported in the SLIM paper for the Lucy model since we used the reference implementation in [Jacobson et al. 2014], which is not using a multi-threaded solver.

triangle area is 10^{-6} in Figure 12). We show an example of one of our iterations in Figure 12, where the highlighted vertex traversed a distance of 150 times the size of the average edge length of its 1-ring in one single step. Using coordinate descent would have required hundreds of iterations to achieve the same effect.

Timings. The timings for all the results in the paper are reported in Table 1.

5 LIMITATIONS AND CONCLUDING REMARKS

We proposed a simple and robust algorithm to generate bijective maps, both in 2D and 3D. We demonstrated the practical value of the algorithm in UV mapping and deformation applications, and its robustness with extensive stress tests.

One major venue for future work is the support of hard positional constraints, which are favored over soft constraints in many practical applications. Our current algorithm only supports soft positional constrains added to the geometric energy [Schüller et al. 2013]. To support hard constraints we would need a guaranteed way to generate a bijective starting point that satisfies those constraints, and then preserve those constraints in our optimization. While bijecive maps with hard constraints can be constructed for a 2D patch homeomorphic to a disk [Weber and Zorin 2014] and for a 3D volume homeomorphic to a sphere [Campen et al. 2016], a generic solution is still elusive.

For 3D cases, the generation of the initial scaffold is not as robust as in 2D, since Tetgen fails for geometries with self-intersections or other imperfections. Our algorithm is also slower since the linear system is larger and denser than 2D and we must use local mesh refinement operations at each step instead of regenerating the entire tetrahedralization. We believe that a more optimized and parallel implementation could reduce this overhead, and we plan to explore this avenue in the future.

REFERENCES

Noam Aigerman, Roi Poranne, and Yaron Lipman. 2014. Lifted Bijections for Low Distortion Surface Mappings. *ACM Trans. Graph.* 33, 4 (2014), 69:1–69:12.

Samantha Ainsley, Etienne Vouga, Eitan Grinspun, and Rasmus Tamstorf. 2012. Speculative Parallel Asynchronous Contact Mechanics. *ACM Trans. Graph.* 31, 6, Article 151 (2012), 8 pages.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013. Integer-grid Maps for Reliable Quad Meshing. *ACM Trans. Graph.* 32, 4, Article 98 (July 2013), 12 pages.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer Quadrangulation. *ACM Trans. Graph.* 28, 3, Article 77 (July 2009), 10 pages.

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

P. Degener, J. Meseth, and R. Klein. 2003. An Adaptable Surface Parameterization Method. In *Proceedings of the 12th International Meshing Roundtable.* 201–213.

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

Michael S. Floater and Kai Hormann. 2005. Surface Parameterization: a Tutorial and Survey. In *In Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization.* Springer Verlag, 157–186.

Xiao-Ming Fu and Yang Liu. 2016. Computing Inversion-free Mappings by Simplex Assembly. *ACM Trans. Graph.* 35, 6, Article 216 (Nov. 2016), 12 pages.

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

Craig Gotsman and Vitaly Surazhsky. 2001. Guaranteed intersection-free polygon morphing. *Computers & Graphics* 25, 1 (2001), 67–75.

David Harmon. 2010. *Robust, efficient, and accurate contact algorithms.* Ph.D. Dissertation. Columbia University.

David Harmon, Daniele Panozzo, Olga Sorkine, and Denis Zorin. 2011. Interference-aware Geometric Modeling. *ACM Trans. Graph.* 30, 6, Article 137 (Dec. 2011), 10 pages.

David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. 2009. Asynchronous Contact Mechanics. *ACM Trans. Graph.* 28, 3, Article 87 (July 2009), 12 pages.

K. Hormann and G. Greiner. 2000. MIPS: An Efficient Global Parametrization Method. In *Curve and Surface Design: Saint-Malo 1999.* 153–162.

Kai Hormann, Bruno Lévy, and Alla Sheffer. 2007. Mesh Parameterization: Theory and Practice. In *ACM SIGGRAPH 2007 Courses (SIGGRAPH '07).* ACM, New York, NY, USA.

Alec Jacobson, Daniele Panozzo, et al. 2014. libigl: A simple C++ geometry processing library. (2014). http://igl.ethz.ch/projects/libigl/.

P. Jimenez, F. Thomas, and C. Torras. 2001. 3D collision detection: a survey. *Computers & Graphics* 25, 2 (2001), 269 – 285.

Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can Mean-Curvature Flow Be Modified to Be Non-singular? *Comput. Graph. Forum* 31, 5 (Aug. 2012), 1745–1754.

Bryan Matthew Klingner. 2009. *Tetrahedral mesh improvement.* Ph.D. Dissertation. University of California at Berkeley.

Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Trans. Graph.* 35, 4, Article 134 (July 2016), 11 pages.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. Graph.* 21, 3 (July 2002), 362–371.

Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Trans. Graph.* 31, 4 (2012), 108:1–108:13.

Yaron Lipman. 2013. Construction of Injective Mappings Of Meshes. *CoRR* abs/1310.0955 (2013). http://arxiv.org/abs/1310.0955

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Proceedings of the Symposium on Geometry Processing (SGP '08).* 1495–1504.

Marek Krzysztof Misztal and Jakob Andreas Bærentzen. 2012. Topology-adaptive Interface Tracking Using the Deformable Simplicial Complex. *ACM Trans. Graph.* 31, 3, Article 24 (June 2012), 12 pages.

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2015. Air Meshes for Robust Collision Handling. *ACM Trans. Graph.* 34, 4, Article 133 (July 2015), 9 pages.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-aligned Global Parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (July 2014), 14 pages.

Kazim Pal, Christian Schüller, Daniele Panozzo, Olga Sorkine-Hornung, and Tim Weyrich. 2014. Content-Aware Surface Parameterization for Interactive Restoration of Historical Documents. *Computer Graphics Forum (proceedings of EUROGRAPHICS issue)* 33, 2 (2014).

Roi Poranne and Yaron Lipman. 2014. Provably Good Planar Mappings. *ACM Trans. Graph.* 33, 4 (2014), 76:1–76:11.

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

Leonardo Sacht, Alec Jacobson, Daniele Panozzo, Christian Schüller, and Olga Sorkine-Hornung. 2013. Consistent Volumetric Discretizations Inside Self-Intersecting Surfaces. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)* 32, 5 (2013), 147–156.

Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. 2001. Texture Mapping Progressive Meshes. In *ACM SIGGRAPH.* 409–416.

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. In *Symposium on Geometry Processing.* 125–135.

Alla Sheffer, Emil Praun, and Kenneth Rose. 2006. Mesh Parameterization Methods and Their Applications. *Found. Trends. Comput. Graph. Vis.* 2, 2 (2006), 105–171.

Jonathan Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. *Applied computational geometry towards geometric engineering* (1996), 203–222.

Hang Si. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)* 41, 2 (2015), 11.

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (July 2015), 9 pages.

Justin Solomon. 2015. *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics.* A. K. Peters, Ltd., Natick, MA, USA.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing (SGP '07).* 109–116.

Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. 2002. Bounded-distortion Piecewise Mesh Parameterization. In *Proceedings of the Conference on Visualization.* 355–362.

W. T. Tutte. 1963. How to draw a Graph. *Proceedings of the London Mathematical Society* 13, 3 (1963), 743–768.

Ofir Weber and Denis Zorin. 2014. Locally Injective Parametrization with Arbitrary Fixed Boundaries. *ACM Trans. Graph.* 33, 4, Article 75 (July 2014), 12 pages.

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2005. Feature-based Surface Parameterization and Texture Mapping. *ACM Trans. Graph.* 24, 1 (Jan. 2005), 1–27.

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: Stretch-driven Mesh Parameterization Using Spectral Analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '04).* ACM, New York, NY, USA, 45–54.