Eurographics Symposium on Geometry Processing 2013
Yaron Lipman and Richard Hao Zhang
(Guest Editors)

Volume 32 (2013), Number 5

Locally Injective Mappings

Christian Schüller[1] Ladislav Kavan[2] Daniele Panozzo[1] Olga Sorkine-Hornung[1]

[1]ETH Zurich, Switzerland
[2]University of Pennsylvania, USA

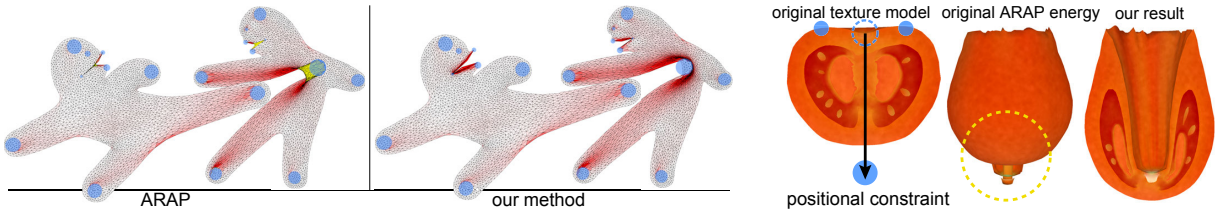

Figure 1: *Popular deformation energies, such as As-Rigid-As-Possible, do not sufficiently penalize element inversion, and "spikes" and "spillages" might occur as a consequence (visualized in yellow). Our method modifies any given deformation energy, such that inversions are infinitely penalized. We derive an efficient numerical method to minimize the modified energy and produce a locally injective mapping of 2D domains (left) and volumetric 3D objects such as Diffusion Surfaces models [TSNI10] (right).*

Abstract

Mappings and deformations are ubiquitous in geometry processing, shape modeling, and animation. Numerous deformation energies have been proposed to tackle problems like mesh parameterization and volumetric deformations. We present an algorithm that modifies any deformation energy to guarantee a locally injective mapping, i.e., without inverted elements. Our formulation can be used to compute continuous planar or volumetric piecewise-linear maps and it uses a barrier term to prevent inverted elements. Differently from previous methods, we carefully design both the barrier term and the associated numerical techniques to be able to provide immediate feedback to the user, enabling interactive manipulation of inversion-free mappings. Stress tests show that our method robustly handles extreme deformations where previous techniques converge very slowly or even fail. We demonstrate that enforcing local injectivity increases fidelity of the results in applications such as shape deformation and parameterization.

1 Introduction

No realistic material can be compressed to zero or even negative volume. While this is a very basic and intuitive requirement, it is rather difficult to satisfy in practice when constructing continuous piecewise linear mappings, as is often done in geometry processing, shape modeling, and animation. This problem arises for example in mesh parameterization, which seeks a mapping between a surface and a subset of $\mathbb{R}^2$, useful for transfering 2D data onto the surface. In some situations this map should be bijective (e.g. for texture mapping) or at least locally injective (e.g. for remeshing). Yet, many common methods for parameterization guarantee neither bijectivity nor local injectivity. In this paper we focus on local injectivity which requires that the determinant of the Jacobian of our mapping is always positive.

We consider both 2D and 3D mappings. Mappings from $\mathbb{R}^3$ to $\mathbb{R}^3$ are typically used to deform objects, and similarly to parameterization, these maps should be locally injective, because inverted elements correspond to physically impossible deformation. In physics-based simulation, it is often sufficient to be able to recover from inverted elements rather than avoid them [ITF04]. While useful in some applications, this approach is not adequate if we want to guarantee that all elements will have positive volume.

A common way of computing a mapping f between arbitrary shapes is by designing an energy functional $E(f)$ that measures the desired properties of the mapping, such as the amount of conformal and area distortion. The mapping is then computed as the minimizer of the energy E under some constraints, for example the desired positions of the

manipulation handles in interactive shape deformation. The main contribution of this paper is a method for efficient, interactive optimization of an arbitrary energy E subject to *hard* non-inversion constraints, i.e., guaranteeing that all elements retain positive volume. Drawing inspiration from barrier methods [BV04], our method works by adding a term which grows to infinity as the area/volume of the deformed elements approaches zero. The resulting energy is highly nonlinear and difficult to optimize; indeed, commercial optimization software [BNW06] often takes impractically long time to converge and effectively precludes user interaction. In this paper, we propose a carefully designed barrier term and the related numerical optimization procedures that allow us to obtain a robust and fast solver. Using our technique, the user can manipulate locally-injective mappings *interactively*.

Recently, Lipman [Lip12] introduced a method that minimizes an arbitrary energy over a space that is guaranteed to contain only locally-injective mappings. This method is applicable only to planar maps and does not offer interactive response already for moderately sized meshes. In contrast, our approach works by augmenting an arbitrary deformation energy, which trivially extends to 3D and, most importantly, permits interactive response times. This is crucial in user-driven shape deformation, because the system needs to provide feedback to the user with a minimal delay. Our method makes this possible thanks to the fact that each step is sufficiently fast and converges smoothly to the solution, enabling the user to quickly detect and resolve problematic edits. Such interactivity may also be useful in parameterization, where the user may prefer to have direct control over the solution.

We apply our method to several existing deformation energies, such as simple Laplacian editing [BS08], As-Rigid-As-Possible (ARAP) [SA07, LZX*08] and a simple linear elasticity model utilizing Green's strain [BW97]. We show that our algorithm fixes the well known spill-overs and sharp protrusion artifacts (spikes), caused by moving a point constraint too far from its original position (Figure 2). Green's strain energy is not commonly used for complicated deformations because it does not penalize inverted elements, often leading to poor deformation quality. With our term added to the energy, this disadvantage disappears and Green's strain produces results similar to the more complex ARAP methods.

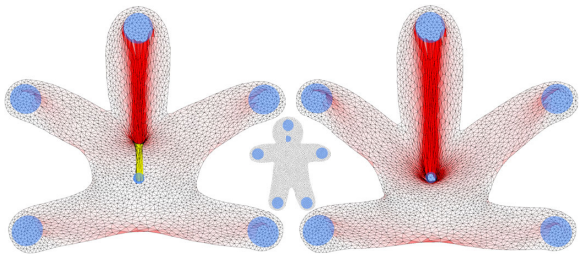

Figure 2: *Example of 2D deformation: (left) the original ARAP energy minimization, (right) our method, (middle) rest pose. Blue circles enclose constrained vertices, red color indicates the amount of angular distortion. Yellow indicates inverted elements.*

We compare our method to standard numerical approaches and barrier functions, showing its superior performance and robustness in the setting of interactive deformations.

2 Related work

Shape deformation and parameterization are typically separately studied problems in geometry processing. However, our technique is equally suitable in both cases, and therefore we survey existing techniques for both.

Deformation. Deformation modeling is important both in geometry processing and physics-based simulation [NMK*06]. Linear deformation models are simple and efficient, but cannot avoid artifacts when large deformations are required [BS08]. This is due to the fact that deformation energies should be invariant to rotation, which is a nonlinear function of the shape geometry. One possibility is to apply nonlinear strain measures, such as Green's strain [BW97]. Unfortunately, Green's strain does not penalize inversion. Another, more costly possibility is to use SVD of the deformation gradient, which allows to detect and resist inversion [ITF04, SHST12]. While penalizing inverted elements, inversion is not guaranteed to be avoided, in contrast to our method. A related idea is to explicitly decompose the deformation gradient into a rotation and pure deformation (stretch) components. This strategy is utilized in shape matching [MHTG05], ARAP deformation energies [SA07, LZX*08], and their continuous limit [CPSS10]. Other methods like [AOW*08, MKN*04] propose to use a penalty term based on the determinant of the deformation gradient to preserve volume. While these methods discourage inverted elements, they do not always avoid them; negative volume elements may and often do occur.

In order to guarantee that element inversion is avoided, we can use a deformation energy that grows to infinity as the area of a deformed element approaches zero. This is clearly impossible with linear deformation energies. Several constitutive models with this property are known in continuum mechanics, such as the Neo-Hookean model [BW97]. While this model is implemented in physics-based simulation software [PMS12, SSB13], it is not used frequently, mainly due to the increased numerical complexity. [vFTS06, AS07] restrict the displacement introduced by the deformation to be a divergence-free vector field, which prevents local and global inversion. Another approach by [AWC04, AC04] decomposes a space deformation into a series of small steps, locally preventing any introduced foldovers. A barrier term based on a distance function is used in [PS06] to prevent crossings of maze paths. The determinant of the deformation gradient is used in [DKMS04] to define an energy which prevents flips. In this paper, we enrich existing energies by adding a carefully designed inversion-avoiding term. We also propose a corresponding nonlinear optimization procedure to achieve smooth convergence and interactive user response even for generously tessellated models.

Parameterization. Single-patch parametrization maps a surface homeomorphic to a disk to a subset of $\mathbb{R}^2$. Here we mainly review the work dealing with locally injective param-

eterization, and we refer an interested reader to [FH05] and [SPR06] for additional references. Tutte's theorem [Tut63] guarantees that a parametrization with fixed convex boundary, where the location of each inner vertex is a convex combination of its neighbors' locations, is bijective. Such parameterization is obtained by solving a sparse linear system [Flo03]. Since this construction cannot be used if the domain is concave, which is often necessary to reduce distortion, alternative parametrization approaches have been proposed. In particular, free-boundary methods, for example methods that minimize the discrete conformal energy (see e.g. [LPRM02, MTAD08]), enable to significantly reduce the distortion compared to fixed boundary. However, no linear free-boundary method can guarantee a fold-over free parametrization. It is possible to iteratively repeat the optimization with penalty terms that fight against flipped triangles [BZK09] but convergence is not guaranteed.

A direct way to prevent inversions is to employ a nonlinear energy that penalizes inversions and degenerate elements by an infinite cost [HG00]. However, such energies are typically hard to optimize since the gradient and the Hessian become ill-conditioned when the elements degenerate. Customized optimization strategies could be used, for example [SSGH01] and [DMK03] propose a coordinate descent optimization where each vertex is moved separately. Similarly, [SLMB05] enforces local injectivity after computing the parametrization by moving a vertex in the kernel of its one-ring and smoothing the resulting parametrization to reduce distortion. Such methods may get stuck and fail to find an un-inverted configuration if the vertex kernels are too small. Global collision detection could also be used to prevent inverted elements [HPSZ11]. However, this method becomes prohibitively slow when the deformation generates multiple collisions.

As Rigid As Possible (ARAP) energies mentioned above have been used for parametrization purposes to obtain as isometric as possible parameterizations [LZX*08, CPSS10]. As in the shape deformation case, ARAP does not guarantee that inversion will be avoided, in fact inverted elements often appear if the parameterization is far from isometry. Some parameterization approaches (e.g. [SCOGL02, LPRM02, MZ12]) split the domain into multiple parts in order to bound the distortion. Inverted elements can still be introduced in the single-patch parametrization step; our algorithm could be used to make the per-patch parameterization locally injective with minimal modifications.

Local injectivity in 2D. Generating 2D locally-injective maps has been studied in two recent papers. [WMZ12] computes, for a fixed boundary, a map with the lowest conformal distortion. The method is limited to 2D maps, requires a fixed boundary, and cannot be applied to arbitrary energies. [Lip12] proposes a representation of the subspace of maps with bounded maximal conformal distortion; these maps by definition are locally injective. An arbitrary energy can be minimized on this subspace, resulting in inversion-free parameterization or 2D deformation. However, to find a solution, the subspace is convexified, which potentially makes the feasible region empty. Finding a solution may also fail if the bound on maximal distortion is set too low (simply because a map with this distortion bound does not exist). The method relies on certain special properties of 2D mappings; an extension to 3D was not discussed and does not seem simple. The method utilizes a quadratic programming solver and allows minimizing the energy with only few iterations, but each iteration might take tens of seconds even for moderately sized meshes, making it too slow for user interaction.

3 Method

Minimization of any deformation energy is not very interesting unless we introduce additional terms that compete with the energy. We can express this by adding a soft constraint term to an arbitrary energy $E(\mathbf{v})$ that depends on the position of the mesh vertices $\mathbf{v}$:

$$\operatorname*{argmin}_{\mathbf{v}} \; E(\mathbf{v}) + \alpha \|\mathbf{C}\mathbf{v} - \mathbf{d}\|^2 \tag{1}$$

where $\alpha \geq 0$ is a parameter specifying the weight of the soft constraints. This parameter is automatically estimated in our algorithm (Section 3.4). Note, that hard constraints may preclude the existence of a non-inverted minimum in certain constraints and mesh configurations, due to the piecewise-linear nature of the meshes.

We propose to modify an existing deformation energy E by adding a barrier term which grows to infinity as our elements approach zero area/volume. We propose a solver which is customized for interactive generation of maps, with a user in the loop. We describe the barrier term in Section 3.1, and the solver we use in Section 3.2. The barrier functions tend to make the Hessian of this augmented energy ill-conditioned in presence of large deformations, and a strategy to prevent this, while also generating a smooth sequence of intermediate optimization results, is presented in Sections 3.3 and 3.4.

3.1 Non-flip constraints as barrier functions

We start with the function $\lambda_j(\mathbf{v})$, which measures the area/volume of the jth element (i.e., triangle in $d = 2$ or tetrahedron in $d = 3$) in configuration $\mathbf{v}$. This function is a fraction of the determinant, i.e., a polynomial of degree d. We define our constraint functions as

$$c_j(\mathbf{v}) = \lambda_j(\mathbf{v}) - \varepsilon \tag{2}$$

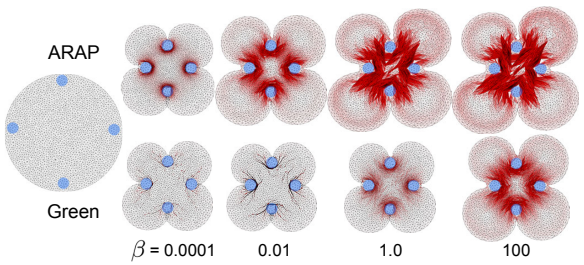

Figure 3: *The influence of the barrier strength β on the optimization for Green's strain and ARAP energies. When β is high, the optimization effectively tries to preserve the original element areas. Note that any $\beta > 0$ prevents flips, since the total energy is infinite for non-positive element area.*

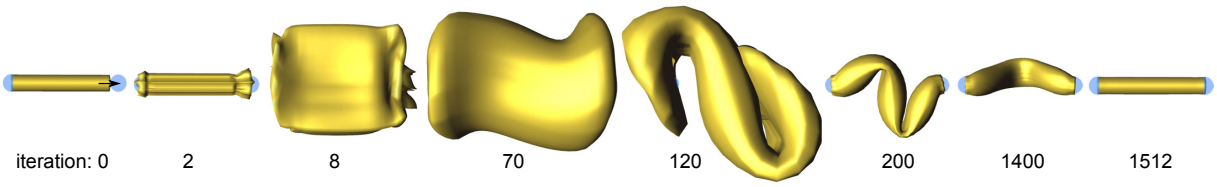

Figure 5: *Visualization of the intermediate iteration states of the interior point method implemented in* KNITRO *[BNW06]. In this simple example we stretch a cylinder by pulling the right cap, while keeping the left cap fixed. Note that the convergence is not monotonic and the intermediate states are unintuitive.*

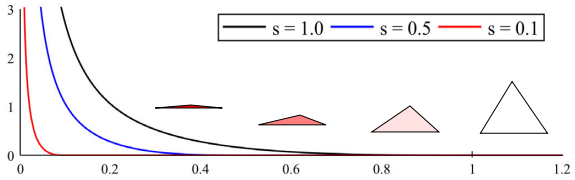

Figure 4: *Plot of the barrier as function of the triangle area, for three different values of the constant s. As the triangle degenerates, the barrier function increases, approaching* ∞.

where ε is a small constant that accounts for numerical inaccuracies; we used $\varepsilon = 10^{-5} \cdot \min_j \lambda_j(\mathbf{v}_0)$ in our experiments, where $\mathbf{v}_0$ is the initial mesh geometry. By requiring $c_j(\mathbf{v}) > 0$, we say that elements cannot invert. One possibility would be to use these constraints directly and solve a nonlinear constrained optimization problem. However, this is complicated, time consuming, and not suitable for interactive manipulation. Instead, we propose to modify the original energy by adding a fixed barrier term. A barrier is a function $\phi : \mathbb{R} \to \mathbb{R}$ such that $\lim_{x\to 0} \phi(x) = \infty$ and for $x \leq 0$, $\phi(x) = \infty$. Adding the barriers yields an unconstrained optimization problem:

$$\underset{\mathbf{v}}{\operatorname{argmin}} \quad E(\mathbf{v}) + \alpha \|\mathbf{C}\mathbf{v} - \mathbf{d}\|^2 + \beta \sum_{j \in \mathcal{E}} \phi_j(c_j(\mathbf{v})) \qquad (3)$$

where $\beta > 0$ is a scalar parameter specifying the barriers strength (Figure 3) and $\mathcal{E}$ is the set of all elements. Numerical issues aside, the minimization of Eq. 3 guarantees that no element may invert regardless of the value of β. Note that we assume that the rest pose $\mathbf{v}_0$ does not contain inverted elements.

We propose a barrier function that only affects the deformation energy when the elements are *close* to being degenerate. In other words, in addition to requiring $\lim_{x\to 0} \phi(x) = \infty$, we want the new barrier function to smoothly approach 0 as $x \to s_j$, where $s_j > 0$ is a certain fraction of the initial rest-pose area/volume of the jth element, i.e., $s_j = s\lambda_j(\mathbf{v}_0)$, $s > 0$. This necessitates defining a different barrier function for each element, but this is just a technical matter.

We build our barrier functions ϕ_j starting from a cubic polynomial g_j, which must possess the following properties:

$$g_j(0) = 0, \;\; g_j(s_j) = 1, \;\; g_j'(s_j) = 0, \;\; g_j''(s_j) = 0 \qquad (4)$$

This results in

$$g_j(x) = \frac{1}{s_j^3}x^3 - \frac{3}{s_j^2}x^2 + \frac{3}{s_j}x. \qquad (5)$$

We define our "compact" barrier functions as "splines", using the inverse of the above polynomial:

$$\phi_j(x) = \begin{cases} \infty, & x \leq 0 \\ \frac{1}{g_j(x)} - 1, & 0 < x < s_j \\ 0, & x \geq s_j \end{cases} \qquad (6)$$

The parameter s (Figure 4) determines how much the area can be reduced before the barriers intervene to prevent inversion. It is tempting to use a very small value for s, so that the barriers affect the energy as little as possible. The catch is that a small s makes the slope of the barrier function very steep, leading to a poor quadratic approximation of the energy, which we need in our numerical optimization (Section 3.2). This reduces the convergence speed. We found that a good compromise between numerical properties and approximation of the original energy is a value of s between $0.1 - 1$.

Discussion. The choice of the barrier function was not straightforward and we first experimented with existing methods, which all turned out to be ill suited for our purposes.

The nonlinear optimization problem in Equation 1 with the non-flipping constraints can be directly solved using an interior point method (IPM). We tested two state-of-the-art solvers, KNITRO [BNW06] and Ipopt [WB06], and we observed that they are not suitable when interactivity is desired, because they converge to the solution in a non-monotonic way. In Figure 5, we show the intermediate steps of KNITRO for a simple deformation example. The entire optimization took 1512 iterations and 948 seconds, compared to 4 iterations and less than 1 second of our solver. This behavior is due to the weights of the barrier term that is modified at every iteration [BV04]. For our setting, we want the energy E to decrease at each iteration, so that we can show the intermediate results of the optimization.

Inspired by interior point methods, we initially used the following *log*-barriers:

$$\phi_{log}(x) = \begin{cases} -\log(x), & \text{if } x > 0 \\ \infty, & \text{otherwise} \end{cases} \qquad (7)$$

While preventing flipped elements, these barriers achieve

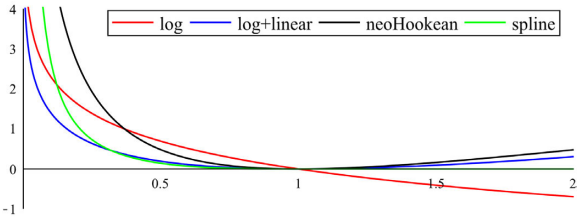

Figure 6: *Barrier functions plots. Only the inverse-spline barrier has compact support and does not affect the stretching of elements.*

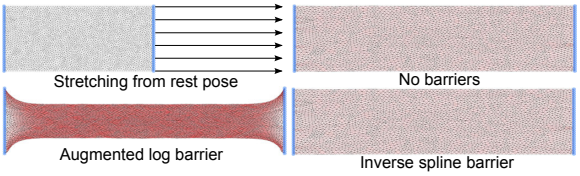

Figure 7: *Visualization of the bias for the augmented log barrier, compared to the inverse-spline function, minimizing Green's strain energy.*

their minimum value only at infinity and arbitrarily alter the rest state of the deformation energy, which is also the reason for the strange intermediate shapes in Figure 5. Therefore, we augment them with an additional linear term to fix this problem. The resulting barrier function will now have the minimum at the rest pose of the deformation energy, but it still alters the energy when elements increase their area/volume. The modified barriers are similar to an energy term found in the model of constitutive Neo-Hookean materials [BW97]:

$$\phi_{hookean}(x)_j = \begin{cases} \log(\frac{x_j}{s_j})^2, & \text{if } x > 0 \\ \infty, & \text{otherwise} \end{cases} \tag{8}$$

Also the Neo-Hookean barrier biases the deformation energy, since it tries to preserve the area of every element and can compete with deformation energies where stretching is allowed (for example in any energy used to compute conformal parametrizations). A visual comparison is given in Figures 6 and 7, energy and timing measurements can be found in Table 1, where our barrier outperforms the other options.

3.2 Optimization

To minimize Equation 3, we use a variant of the Levenberg-Marquardt (LM) algorithm [GW81]. The barrier terms offer a relatively simple closed form of the gradient and Hessian, see the additional material. The Hessian $\nabla^2 E_{bar}(\mathbf{v})$ is used to quadratically approximate the energy and determine a direction to advance, decreasing the energy at every iteration and converging to a local minimum. We iteratively update the vertex positions in two steps:

$$\begin{aligned} \mathbf{p}_i &= (\nabla^2 E_{bar}(\mathbf{v}_i) + \mu_i \mathbf{I})^{-1} \cdot \nabla E_{bar}(\mathbf{v}_i) \\ \mathbf{v}_{i+1} &= \mathbf{v}_i - \sigma_i \mathbf{p}_i \end{aligned} \tag{9}$$

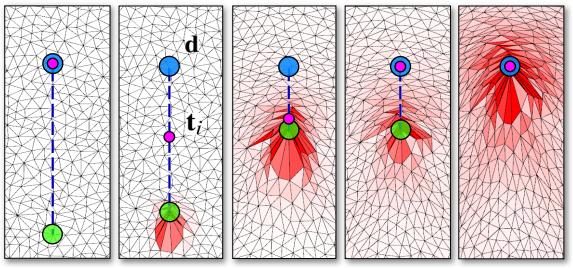

Figure 8: *A close-up of a mesh during the substepping iterations of our algorithm. The target position* $\mathbf{d}$ *(blue disc) of the constrained vertex (green disc) is linearly interpolated to define the intermediate target* $\mathbf{t}_i$ *(magenta disc). As the shapes of the elements improve, the relaxation is tightened and converges to the prescribed positional constraint.*

where σ_i is the step size and is computed with an adaptive, backtracking line search method: we use an initial and maximum value of 1 for σ_i and we halve until we find a step size that decreases the energy. At ith step, we use σ_{i-1} as a starting point, and multiply it by two if the step size decreases the energy. We experimented with more advanced line search methods without getting significant benefit, since the time saved by the reduced number of iterations was compensated by the increased number of energy evaluations needed.

The Hessian of our energy becomes numerically singular when an element degenerates, and then it is impossible to solve the linear system in Equation 9 with $\mu_i = 0$. To ensure we always find a search direction, we regularize the Hessian by adding $\mu_i \mathbf{I}$. The factor μ_i is chosen to make the Hessian invertible, while being as small as possible and eventually zero if no regularization is needed. It is dynamically updated using the same strategy we used for σ_i, where the condition we check is the invertibility of the regularized Hessian (using sparse Cholesky decomposition). The factor μ_i has a very intuitive effect: for a small or large Hessian the step direction $\mathbf{p}_i$ becomes more similar to the gradient descent with a step size $1/\mu_i$. This results in slower but more reliable convergence whenever a good quadratic approximation of the energy is not available.

3.3 Substepping

The solver, presented in Section 3.2, minimizes the energy in Equation 3 only for moderate deformations. For extreme cases like in Figure 2, this method may fail to make any reasonable progress. We propose a novel substepping strategy customized for our specific problem.

The core idea is to rely on the search direction provided by the Hessian when it is well-conditioned and alter our objective function when it is not. The barrier makes the Hessian numerically singular in configurations similar to Figure 8, where a group of consecutive triangles is compressed by a positional constraint that forces them to degenerate. Relying on the search direction provided by the Hessian will force the algorithm to perform many small iterations or even stop due to rounding errors. We thus propose to relax the positional con-

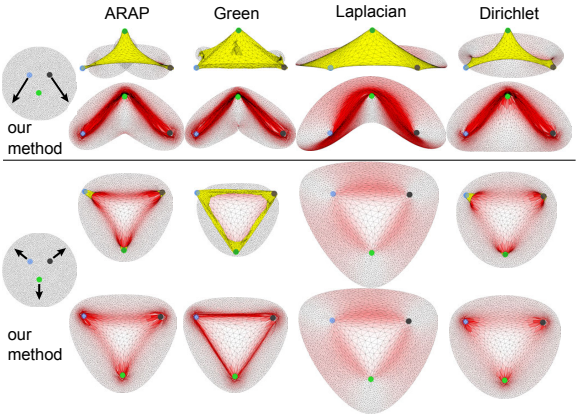

Figure 9: *Deformation of a 2D mesh using different energies. First and third rows show the results of the unmodified energy minimization, and second and fourth rows display our results.*

straints to allow the deformation energy and the barrier term to improve the shape of the degenerate elements. We modify the positional constraints by replacing the user-provided values $\mathbf{d}$ with intermediate targets $\mathbf{t}_i$ updated at every step:

$$E_{bar}(\mathbf{v}) = E(\mathbf{v}) + \alpha \|\mathbf{C}\mathbf{v} - \mathbf{t}_i\|^2 + \beta \sum_{j \in \mathcal{E}} \phi_j(c_j(\mathbf{v})) \quad (10)$$

$\mathbf{t}_i$ is identical to $\mathbf{d}$ if the Hessian is invertible, and is modified to be further from $\mathbf{d}$ depending on the amount of regularization needed to make the Hessian non-singular. We want the positional constraints term to gradually disappear for large μ_i and be equal to $\mathbf{d}$ for $\mu_i = 0$. This way, the optimization is affected only if the Hessian is ill-conditioned:

$$\mathbf{t}_i = \mathbf{C}\mathbf{v}_{i-1} + \frac{1}{1+\mu_i^2}(\mathbf{d} - \mathbf{C}\mathbf{v}_{i-1}) \quad (11)$$

We experimentally found that squaring μ_i works well, in particular because $\mathbf{t}_i$ approaches $\mathbf{d}$ super-linearly in μ_i. Introducing $\mathbf{t}_i$ can be interpreted as an adaptive "substepping" method. This relaxation allows the barriers to improve the shape of the elements, resulting in gradually decreasing μ_i, which will in turn tighten the relaxation and eventually bring $\mathbf{t}_i$ to $\mathbf{d}$. We show a series of iterations of this algorithm in Figure 8.

3.4 Soft constraints

The weight α (Equation 1) plays an important role in the energy, since it controls the effect of the soft constraints. An incorrect choice of this parameter can result in not satisfying the constraints (if α is too small, see Figure 21), or to converge too early due to numerical issues (if α is too high). Relying on the fact that the positional constraints should always be dominating the energy by a certain ratio r, we introduce an adaptive adjustment of α:

$$\gamma = \frac{r}{\|\mathbf{C}\mathbf{v} - \mathbf{d}\|^2} \left(E(\mathbf{v}) + \beta \sum_{j \in \mathcal{E}} \phi_j(c_j(\mathbf{v})) \right) \quad (12)$$

$$\alpha_{i+1} = \min\left(\max\left(\alpha_i, \gamma\right), t\right) \quad (13)$$

We experimentally found that a value of $r = 1000$ and $t = 10^{16}$ works well for all the examples in the paper. In our experiments, the positional constraint are satisfied up to numerical precision even for extreme deformations. For the stress tests we provide the residuals in the captions of Figure 19 and 20.

4 Results

We ran all our experiments on a single core of an Intel i7 processor clocked at 2.93 GHz. The linear solves inside the LM method are performed using the Cholesky solver of Eigen [GJ*12]. Note that the matrix nonzero pattern stays the same throughout the iterations; we can therefore save some computation time by precomputing the symbolic factorization and perform just the numeric factorization in each iteration. We applied our method to 2D and 3D deformations and mesh parameterization, showing that the quality of the resulting mappings generally improves compared to the original energy minimization without our barriers, while still being computed at an interactive rate. We also performed experiments with extreme deformation, to show the robustness of our approach. Additional results are available in the supplemental material and the accompanying video. In all our results we set the barrier parameter $\beta = 0.01$ for all energies except Green's strain, for which we use $\beta = 100$ (since this energy does not inherently penalize inversions at all, see Figure 3). The ratio s is always set to 1. When minimizing the ARAP energy, we use the local-global approach of [SA07, LZX*08] and apply the barriers in the global step.

Dirichlet/Laplacian. In Figure 10, we show that even a simple deformation, computed using the Dirichlet energy, can introduce inverted elements. The problem is less prominent if the Laplacian energy is used instead.

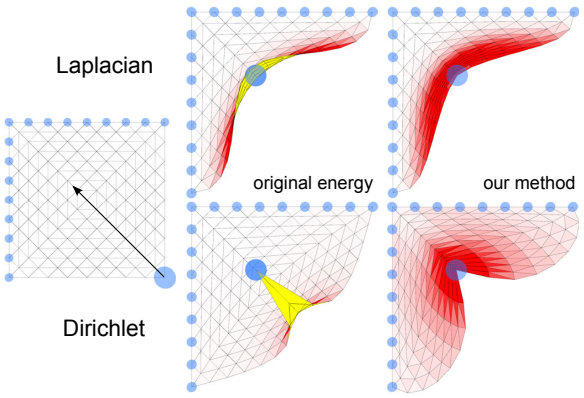

Figure 10: *A simple example of a 2D deformation using the Dirichlet and Laplacian energies, with and without our barrier method.*

2D shape editing. Figure 9 shows simple cases where the Dirichlet, biharmonic (Laplacian), Green's strain and ARAP energies generate inverted elements. This issue arises often (see also Figures 2, 13, 19), and it is fixed when the energy is

augmented with our barriers, generating more intuitive deformations. In all our illustrations, the colored circles enclose the handle vertices, which can be moved interactively. The meshes are colored on a linear scale of the conformal distortion measure proposed in [LZX*08]; inverted elements are colored in yellow. Without barriers, inversion and "spillage" artifacts are introduced for extreme deformations, while our results introduce more distortion around the handles, but do not spill outside of the shape or invert elements. Global self-intersections of the mesh can still occur when the boundary is not fixed, since we do not perform any self-collision detection. Note that with a fixed (simple) boundary, our method produces bijective maps (Figure 17).

Image warping. Our method can also be used to deform 2D images interactively, as shown in Figure 12 and in the accompanying video. Note that if the boundary of the image is simple and fixed, we are guaranteed to produce a bijective deformation.

3D deformations. Unlike previous works on locally-injective mappings [Lip12, WMZ12], our formulation easily extends to 3D deformations. In Figure 11 we compute a natural-looking bend of the cylinder using Green's strain energy, which collapses the shapes when minimized without applying our barriers. During the interactive editing of the handles, the deformation is usually smooth, as shown in the accompanying video. However, for extreme deformations like the one shown in Figure 20, the substepping slows the optimization down, and it might take a few seconds to converge. Figure 13 shows another 3D deformation of the *Dino* model.

We also tested our method on diffusion surfaces [TSNI10], where a locally injective deformation prevents artifacts in the generated volumetric texture, see the deformed tomato in Figure 14. The interior structure of the tomato is deformed without introducing flips or spill-overs, in contrast to the normal ARAP deformation. The editing has been done interactively with a grid resolution of 2250 voxels and 13500 tetrahedra. Anticipating multigrid implementations in the future, we discretize the deformation field using a regular voxel grid; however, the energy is still evaluated on tetrahedra by subdividing each voxel into 6 tets.

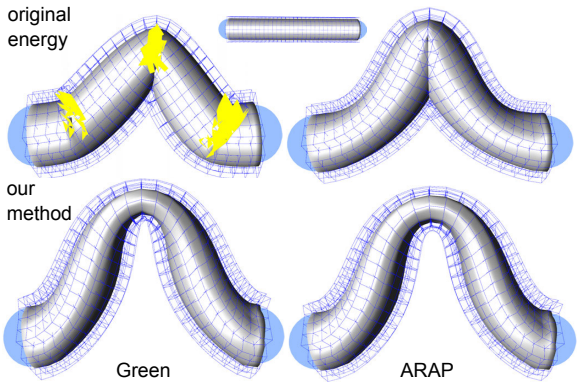

Figure 11: *3D deformation examples using Green's strain and ARAP energies. Note that the deformation is computed on an enclosing volumetric grid and then interpolated onto the cylinder shape itself in this case.*

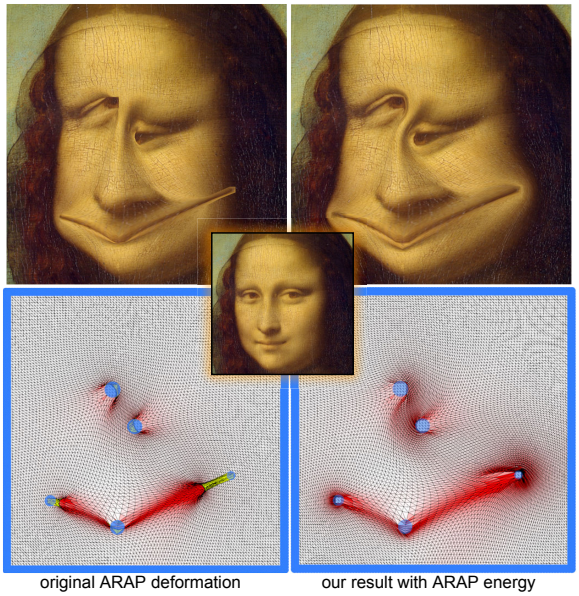

Figure 12: *Image warping with the ARAP energy on a textured regular mesh, with constrained boundary in blue.*

Single-patch parameterization. Local injectivity is particularly important for mesh parameterization, and existing parameterization methods can be modified by our method to guarantee this property. However, our method requires a starting point that is free of self-intersections: if this is not the case, our energy is infinite and we cannot advance. We thus use Tutte's parameterization (with uniform weights), fixing the boundary to a circle as a starting point for all the examples. This parameterization usually contains extreme area distortion, and a complex deformation is required to deform this initial state into a high-quality parameterization. We show in Figure 15 that our method can robustly optimize the ARAP energy [LZX*08].

In all parameterization examples, the boundary is not fixed

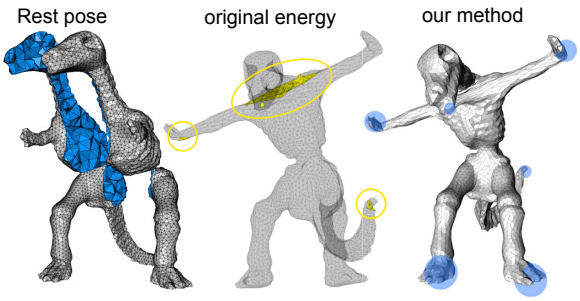

Figure 13: *An example of 3D deformation. A tetrahedral mesh is deformed minimizing Green's strain energy with and without our barriers. The inverted tetrahedra are visualized in yellow.*

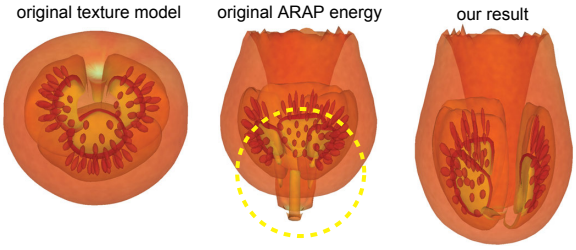

Figure 14: *Deforming a Diffusion Surface, a representation of inner volumetric structure and solid texture [TSNI10]. The setup of the handles is shown in Figure 1, as well as the rendered results of cut-planes with diffused colors. Here we show the same result with transparent rendering of the inner surfaces, which are not intersecting thanks to our barriers.*

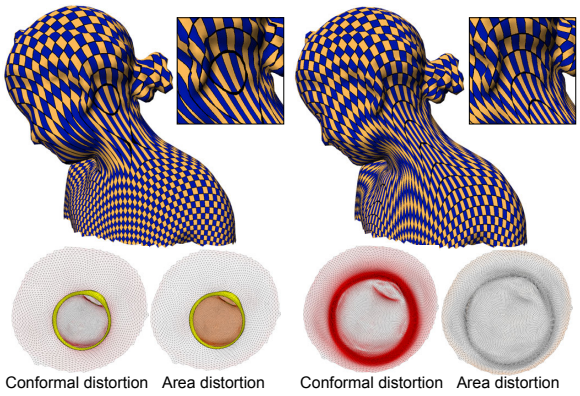

Figure 15: *An example of local parameterization with ARAP, with and without barriers. Red visualizes high conformal distortion and orange the area distortion.*

(only two vertices are fixed to remove the translational and rotational degrees of freedom). Note that our parameter s in fact controls the area distortion, and increasing s tends to preserve the original area of each triangle. As visualized in Figure 15, our method achieves lower area distortion than the original ARAP in this case. Our method allows to interactively edit the parameterization, as shown in Figure 16, where we applied the barriers to the least-squares conformal energy [LPRM02, MTAD08]. By moving two anchors, it is possible to remove global overlaps, making the parameterization bijective. The full editing session, starting from Tutte's mapping, is shown in the accompanying video.

Our method can be used for constrained texture mapping in order to match geometric features to corresponding parts inside the texture image by deforming an existing parameterization. In Figure 17 we show such a deformed parametrization map of a human head in order to align it with the image of a tiger. Even though an extreme deformation is required, our method prevents inversions and produces a plausible, bijective texturing result.

Stress tests. Since we rely on nonlinear optimization with non-convex constraints, we are not guaranteed to find the

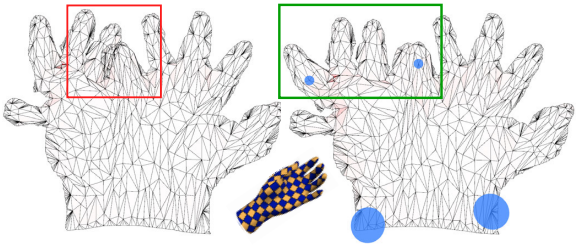

Figure 16: *Interactive removal of global self-intersections using our method. We first computed the parameterization on the left by using our method with the least-squares conformal energy (LSCM [LPRM02]). The result has no inversions, but the two parts of the thumb globally overlap. We then place a few handles and quickly deform the flattened model to resolve the overlaps. See also the accompanying video.*

global minimum of our energy. Additionally, our problem becomes ill-conditioned when triangles start to degenerate, making the optimization harder. However, the combination of our barriers and our substepping seem to successfully alleviate this problem. We provide experimental validation of the robustness of our method, showing that we are able to compute extreme deformations. In Figure 19 we swapped the handles of the arms and the legs of the *Woody* mesh while keeping the head fixed. Our algorithm finds a solution that satisfies the constraints and does not contain any inverted triangles, while ARAP generates a deformation with 512 flipped triangles. We also tested our method on an extreme 3D deformation by twisting a cylinder 7 times (Figure 20). We used a very coarse grid and set the barrier constant $\beta = 0.01$ to make finding a solution harder, and even so, our algorithm computes an extreme deformation without inverted elements. Without barriers, 135 inverted tetrahedra are introduced.

Measurements and timings. The mappings are computed at interactive rates with our solver, as shown in the accompanying video. The extreme ARAP deformation in Figure 2 takes 5.5 seconds for a mesh with 5040 triangles, and the complex texture deformation in Figure 17 (46384 triangles) converges in 172 seconds. The ARAP parametrization of the *Bimba* model (11253 triangles) requires 93 iterations and 16.3 seconds, when initialized with Tutte's parametrization. The 3D deformation in Figure 13 involves 13052 tetrahedra and converges in 26 iterations and 7.3 seconds. The volumetric deformation in Figure 14 is more challenging, since it uses a mesh with 13K tetrahedra and is computed in 208 seconds and 112 iterations. Note that the intermediate iteration results of our method are *plausible* because the convergence is smooth, and thus we can constantly provide interactive visual feedback to the user, so that she can gauge progress and if necessary, interrupt the optimization even before it has fully converged. This is in contrast to interior point methods discussed earlier, whose intermediate states are not useful in intuitively estimating the final result (Figure 5). The time to complete a single iteration of our method is quite short, allowing constant update of the rendering (please refer to the accompanying video). This is different from the method

of [Lip12], which may require only very few iterations to find a solution, but each iteration is typically expensive. For instance, in the bottom row of Figure 18, [Lip12] takes just 2 iterations to find a solution, but each iteration takes 19.5 seconds. Our method converges in 154 iterations and takes a total of 6.5 seconds.

In Table 1 we compare between the different barriers and interior point methods we experimented with. We do this based on two deformations of meshes with different resolutions (Figure 21) minimizing Green's strain energy. We report the number of iterations, time, deformation energy and positional constraint energy values after the convergence of the LM method. A fair comparison of these different barrier terms requires setting the β appropriately: with the same β, the augmented log and Neo-Hookean barriers penalize changes of area less than our inverse spline, leading to significantly different results (see Figure 3). In terms of time and number of iterations, the inverse-spline barrier with substepping and α-update performs best. Note that for the high resolution mesh, many methods without substepping enabled converged in only a few steps without satisfying the positional constraints. As expected, the inverse spline function also has the lowest bias in the deformation energy compared to the Neo-Hookean or augmented log barriers. Although the interior point methods find a better local minimum, they are much slower or do not converge at all in a reasonable time. Enabling the automatic α-update drastically decreases the positional constraint error. Finally, two different failure cases are shown in Figure 21 in the right column: (top) without α adjustment the preset soft constraints are too weak for the Neo-Hookean energy and the method does not converge; (bottom) disabling substepping causes the solver to converge without reaching the positional constraints.

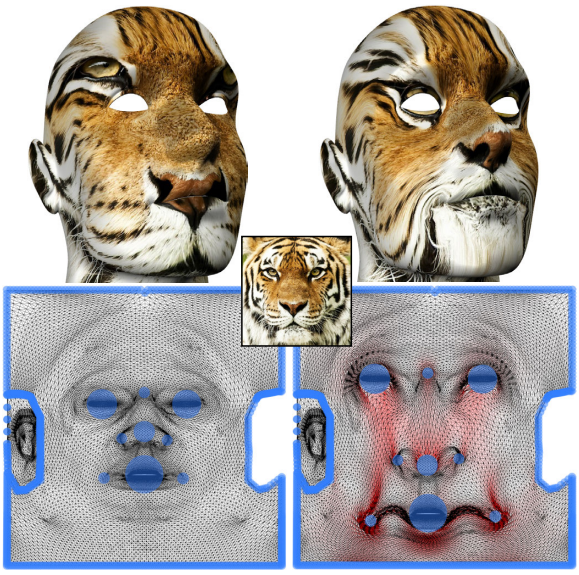

Figure 17: *Texture deformation using the ARAP energy with our method. We fix the boundary of the uv-map and deform inner parts to match the geometric features of the head to the image of the tiger. No inversions occur, and for the fixed boundary case our method generates a bijective mapping.*

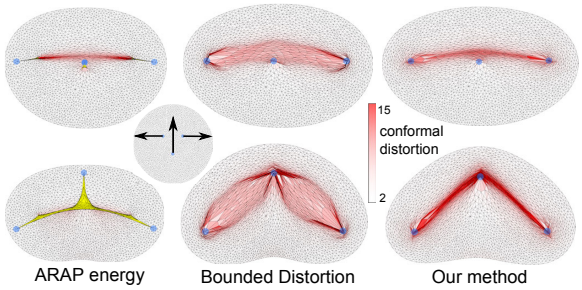

Figure 18: *Comparison of our method to the to Bounded Distortion technique of [Lip12]. The prescribed deformation of the disk is shown in the left inset (we pull the bottom handle up while dragging the other two outwards). Top and bottom rows show the results for smaller and larger handle translation, respectively. The original ARAP energy leads to "spikes", as expected. Our method results in a maximal conformal distortion of* 69 *(top) and* 149 *(bottom) and we used these bounds for Lipman's method. The results in the middle are produced using the author's software. We used the same initial mesh as a starting point for both methods. The distortion color scale is set to* $(2-15)$*, clamping higher values to red; clearly, our method concentrates the distortion around the handles, while Lipman's method distributes it more evenly.*

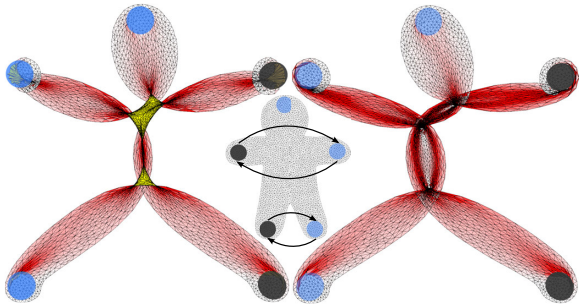

Figure 19: *A 2D stress test: extreme deformation of the* Woody *via ARAP energy minimization, without (left) and with (right) the barriers. Note that although our method cannot avoid global overlaps in this case, no elements are locally inverted. (soft constraint error:* $2.6 \cdot 10^{-16}$*)*

5 Limitations and future work

Our method guarantees that no flipped elements are introduced, but it needs a starting point that also has this property. This is usually not a problem for shape deformation, where the rest pose is normally free from inverted elements. For parameterization, this forces us to use a convex mapping as a starting point. Our method provides only limited control over the change in the deformation energy introduced by the barrier terms. The preservation of the initial area or volume can be controlled with the parameters β and ε. Furthermore,

	Deformation in Fig. 21: 4844 triangles				Deformation in Fig. 21: 48881 triangles			
Barrier & method	**#iter**	**time**	**E · 10^5**	**Error**	**#iter**	**time**	**E · 10^6**	**Error**
IPM (Ipopt)	1136	1636s	1.26	$4.0 \cdot 10^{-06}$		>20h		not converged
IPM (KNitro)	527	68s	1.24	$4.2 \cdot 10^{-06}$		>20h		not converged
Augmented log	309	12s	1.24	$3.6 \cdot 10^{-06}$	69	69s	0.08	$2.8 \cdot 10^{15}$
+ substepping	56	2.8s	1.31	$3.7 \cdot 10^{-06}$	972	831s	75.4	$2.7 \cdot 10^{-08}$
+ α-update	56	2.7s	1.31	$3.7 \cdot 10^{-22}$	555	503s	75.3	$2.5 \cdot 10^{-16}$
Neo-Hookean	2306	96s	6.72	$1.0 \cdot 10^{-04}$	19	15s	0.02	$2.9 \cdot 10^{15}$
+ substepping		>20h		not converged	17917	7.5h	46.2	$3.1 \cdot 10^{2}$
+ α-update	4779	238s	3.40	$2.4 \cdot 10^{-21}$	25814	5.0h	175	$3.5 \cdot 10^{-1}$
Inverse-Spline	332	12s	1.22	$3.5 \cdot 10^{-06}$	150	145s	0.52	$1.9 \cdot 10^{15}$
+ substepping	53	2.3s	1.28	$3.6 \cdot 10^{-06}$	614	517s	63.6	$4.0 \cdot 10^{-02}$
+ α-update	49	1.9s	1.29	$3.6 \cdot 10^{-22}$	526	471s	75.1	$2.9 \cdot 10^{-16}$

Table 1: *Comparison of different barriers and interior point methods on a deformation example from Figure 21, minimizing Green's strain energy on two different resolution meshes. E denotes the energy value and* Error *the soft constraints error.*

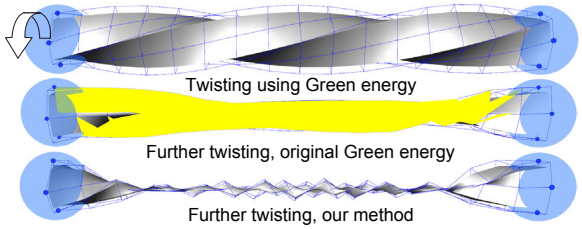

Figure 20: *A 3D stress test: twisting the left handle on the bar 7 times and minimizing Green's strain energy, with and without the barriers. See also the accompanying video. (soft constraint error:* $2.6 \cdot 10^{-26}$*)*

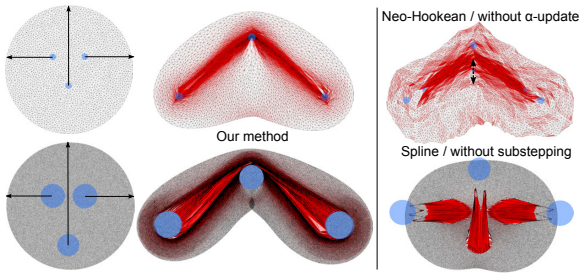

Figure 21: *Deformation of two meshes with 4844 triangles (top) and 48881 triangles (bottom) used for the comparison in Table 1. Right column shows two failure cases for disabled substepping and α-update.*

some deformation energies may require tuning the parameter β, as discussed in Section 4 and Figure 3. Adding our non-flipping term to a deformation energy does not alter the rest pose but it might change the local and global minima when new positional constraints are posed. Additionally, we focus solely on local injectivity. To resolve global overlaps, we could combine our method with self-collision avoidance techniques [HPSZ11].

We plan to further investigate more efficient solver implementations, leveraging parallelism and/or multigrid methods, to improve the performance for highly detailed meshes. In this work, we only utilized soft positional constraints; extension to hard constraints is theoretically simple, but complicated by the fact that a solution may not exist. Another interesting future work would be to implement upper bounds on the maximal conformal distortion [Lip12] using barrier functions. It would also be interesting to introduce adaptive remeshing to mantain high element quality during the optimization.

6 Conclusion

We presented a robust and practical algorithm to generate locally injective 2D and 3D mappings. It can be applied to a variety of deformation energies, preventing the well-known "spike" and "spillage" artifacts. The method can handle extreme deformations and is fast enough to generate interactive results for meshes of moderate sizes. We show that our algorithm can be applied to compute interactive 2D and 3D mesh deformations and also to generate locally injective parameterizations. We hope this work will not only make locally injective mappings practical, but will also promote further research of advanced nonlinear models and numerical methods in geometry processing, shape modeling, and animation.

Acknowledgements

The authors thank Kenshi Takayama for providing the binaries and source code of the Diffusion Surfaces modeling technique [TSNI10]. We are also grateful to Yaron Lipman for sharing the MATLAB implementation of his Bounded Distortion method [Lip12]. This work was supported in part by the ERC grant iModel (StG-2012-306877).

References

[AC04] ANGELIDIS A., CANI M.-P.: Swirling-sweepers: Constant-volume modeling. In *Proc. Pacific Graphics* (2004). 2

[AOW*08] ADAMS B., OVSJANIKOV M., WAND M., SEIDEL H.-P., GUIBAS L. J.: Meshless modeling of deformable shapes and their motion. In *Proc. SCA* (2008). 2

[AS07] ANGELIDIS A., SINGH K.: Kinodynamic skinning using volume-preserving deformations. In *Proc. SCA* (2007). 2

[AWC04] ANGELIDIS A., WYVILL G., CANI M.-P.: Sweepers: Swept user-defined tools for modeling by deformation. In *Proc. SMI* (2004). 2

[BNW06] BYRD R. H., NOCEDAL J., WALTZ R. A.: KNITRO: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization*, vol. 83 of *Nonconvex Optimization and Its Applications*. Springer US, 2006. 2, 4

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph. 14*, 1 (2008). 2

[BV04] BOYD S., VANDENBERGHE L.: *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 2, 4

[BW97] BONET J., WOOD R.: *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997. 2, 5

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009). 3

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 4 (2010). 2, 3

[DKMS04] DIACHIN L. F., KNUPP P. M., MUNSON T. S., SHONTZ S. M.: A comparison of inexact Newton and coordinate descent mesh optimization techniques. In *Proc. International Meshing Roundable* (2004), pp. 243–254. 2

[DMK03] DEGENER P., MESETH J., KLEIN R.: An adaptable surface parameterization method. In *Proc. International Meshing Roundtable* (2003). 3

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling* (2005), Springer. 3

[Flo03] FLOATER M. S.: One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation 72*, 242 (2003), 685–696. 3

[GJ*12] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2012. 6

[GW81] GILL P. R.; MURRAY W., WRIGHT M. H.: *Practical Optimization*. Classics in Applied Mathematics. Academic Press, London, 1981. 5

[HG00] HORMANN K., GREINER G.: MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, Innovations in Applied Mathematics. Vanderbilt University Press, 2000. 3

[HPSZ11] HARMON D., PANOZZO D., SORKINE O., ZORIN D.: Interference aware geometric modeling. *ACM Trans. Graph. 30*, 6 (2011). 3, 10

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. SCA* (2004), pp. 131–140. 1, 2

[Lip12] LIPMAN Y.: Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4 (2012). 2, 3, 7, 9, 10

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (2002). 3, 8

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Comput. Graph. Forum 27*, 5 (2008). 2, 3, 6, 7

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005). 2

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proc. SCA* (2004), pp. 141–151. 2

[MTAD08] MULLEN P., TONG Y., ALLIEZ P., DESBRUN M.: Spectral conformal parameterization. *Comput. Graph. Forum 27*, 5 (2008). 3, 8

[MZ12] MYLES A., ZORIN D.: Global parametrization by incremental flattening. *ACM Trans. Graph. 31*, 4 (2012). 3

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Comput. Graph. Forum 25*, 4 (2006). 2

[PMS12] PATTERSON T., MITCHELL N., SIFAKIS E.: Simulation of complex nonlinear elastic bodies using lattice deformers. *ACM Trans. Graph. 31*, 6 (2012). 2

[PS06] PEDERSEN H. K., SINGH K.: Organic labyrinths and mazes. In *NPAR* (2006), pp. 79–86. 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. Symposium on Geometry Processing* (2007). 2, 6

[SCOGL02] SORKINE O., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *Proc. IEEE Visualization* (2002). 3

[SHST12] STOMAKHIN A., HOWES R., SCHROEDER C., TERAN J. M.: Energetically consistent invertible elasticity. In *Proc. SCA* (2012). 2

[SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: ABF++: fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2 (2005). 3

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (2006). 3

[SSB13] SIN F. S., SCHROEDER D., BARBIC J.: Vega: Non-linear FEM deformable object simulator. *Comput. Graph. Forum 32*, 1 (2013). 2

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *Proc. ACM SIGGRAPH* (2001). 3

[TSNI10] TAKAYAMA K., SORKINE O., NEALEN A., IGARASHI T.: Volumetric modeling with diffusion surfaces. *ACM Trans. Graph. 29*, 6 (2010). 1, 7, 8, 10

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society 13*, 1 (1963). 3

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. Graph. 25*, 3 (2006), 1118–1125. 2

[WB06] WÄCHTER A., BIEGLER L. T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*, 1 (2006), 25–57. 4

[WMZ12] WEBER O., MYLES A., ZORIN D.: Computing extremal quasiconformal maps. *Comput. Graph. Forum 31*, 5 (2012). 3, 7