



HAL
open science

Foldover-free maps in 50 lines of code

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais,
Nicolas Ray, Dmitry Sokolov

► **To cite this version:**

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, et al..
Foldover-free maps in 50 lines of code. 2021. hal-03127350v1

HAL Id: hal-03127350

<https://hal.science/hal-03127350v1>

Preprint submitted on 4 Feb 2021 (v1), last revised 6 Sep 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Foldover-free maps in 50 lines of code

VLADIMIR GARANZHA*, IGOR KAPORIN*, and LIUDMILA KUDRYAVTSEVA*, Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia, Moscow Institute of Physics and Technology, Moscow, Russia
FRANÇOIS PROTAIS†, NICOLAS RAY, and DMITRY SOKOLOV, Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

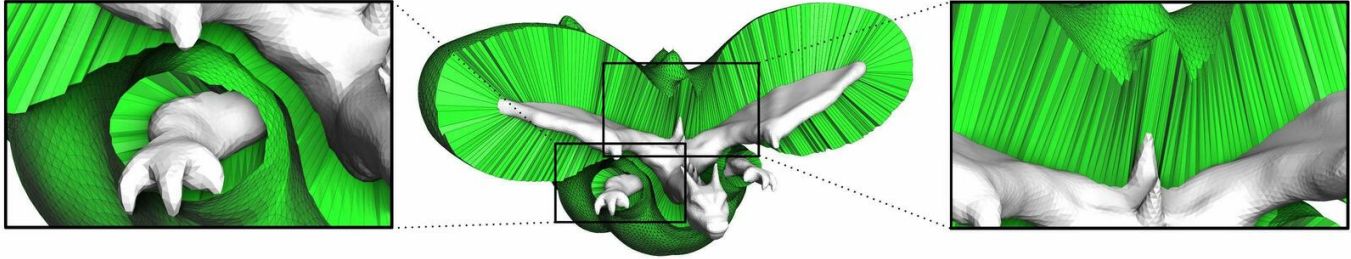


Fig. 1. Our method of constructing injective maps opens a door for a large variety of applications. This figure shows an example of a thick prismatic mesh layer (shown in green) built around a triangulated surface, a very challenging problem for highly curved objects. Thanks to our method, we are able to compute such a layer free of folds and self-intersections.

Mapping a triangulated surface to 2D space (or a tetrahedral mesh to 3D space) is the most fundamental problem in geometry processing. In computational physics, untangling plays an important role in mesh generation: it takes a mesh as an input, and moves the vertices to get rid of foldovers. In fact, mesh untangling can be considered as a special case of mapping where the geometry of the object is to be defined in the map space and the geometric domain is not explicit, supposing that each element is regular. In this paper, we propose a mapping method inspired by the untangling problem and compare its performance to the state of the art. The main advantage of our method is that the untangling aims at producing locally injective maps, which is the major challenge of mapping. In practice, our method produces locally injective maps in very difficult settings, and with less distortion than the previous work, both in 2D and 3D. We demonstrate it on a large reference database as well as on more difficult stress tests. For a better reproducibility, we publish the code in Python for a basic evaluation, and in C++ for more advanced applications.

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

Additional Key Words and Phrases: Parameterization, injective mapping, mesh untangling

1 INTRODUCTION

Most geometric objects are represented by a triangulated surface or a tetrahedral mesh. The mapping problem consists in generating a 2D or 3D map of these objects. This is a fundamental problem of computer graphics because it is much easier for many applications to work in this map space than to directly manipulate the object itself. To give few examples, texture mapping stores colors of a surface as images in the map space, remeshing uses global maps in 2D [Bommes et al. 2013] and 3D [Gregson et al. 2011; Nieser

et al. 2011]. In addition, mapping algorithms can be used to deform volumes [Li et al. 2020], or generate shells from surfaces [Jiang et al. 2020].

What is a good map? Most often maps are represented by the position of the vertices in the map space, and interpolated linearly on each element (triangle or tetrahedron). In a perfect world, the map space would keep the geodesic distances of the object. Unfortunately, this is usually impossible due to Gaussian curvature, and application-specific constraints such as constrained position of vertices or overlaps in the map space. Therefore, the objective of the mapping algorithms is to minimize the distortion between geometric and map spaces, opening the door for numerical optimization approaches as detailed in the survey [Hormann et al. 2008].

What about the invertibility? Unfortunately, when high distortion is required to satisfy the constraints, these algorithms may lose the fundamental property of a maps: injectivity. A solution to preserve it [Floater 1997] relies on Tutte’s theorem [Tutte 1963], however the surface boundary must be mapped to a convex polygon. Despite this strong limitation, it still remains the reference algorithm to generate injective maps. Lower distortion can be obtained by changing weights of the barycentric coordinates [Eck et al. 1995] (as long as they are not negative), and alternative solutions [Campen et al. 2016; Shen et al. 2019] have been explored to improve robustness to numerical imprecision by modifying the mesh connectivity.

Local invertibility. In many applications, maps are used to access a neighborhood of a point within a coherent local coordinate system. To this end, global injectivity is not required, and we instead look for local injectivity [Schüller et al. 2013; Smith and Schaefer 2015]. Their approach starts from an injective map [Floater 1997], and maintains the local injectivity when minimizing the distortion and enforcing the constraints. This allows them to optimize at the same

*This work is supported by the Ministry of Science and Higher Education of the Russian Federation, project No 075-15-2020-799

†Corresponding author: francois.protais@inria.fr

Authors are listed in alphabetical order.

time the parameterization and the texture packing [Jiang et al. 2017], with a possibility to scale to larger meshes [Rabinovich et al. 2017].

Recover local injectivity. Local injectivity can also be recovered for a map with few foldovers present. For example, in 2D [Lipman 2012] and 3D [Aigerman and Lipman 2013], the map is projected on a class of bounded distortion maps. The numerical method is however unlikely to succeed for stiff problems. Recovering local injectivity is also known as mesh untangling. Originally related to Arbitrary Lagrangian-Eulerian moving mesh approach, the mesh untangling problem considers a simplicial complex with misoriented elements and attempts to flip them back by optimizing the position of the vertices. There is an abundant literature on mesh untangling [Du et al. 2020; Escobar et al. 2003; Freitag and Plassmann 2000; Knupp 2001; Toulorge et al. 2013], however the common opinion is that untangling is a very hard problem and algorithms are not robust enough. As a manifestation of frustration over this problem [Danczyk and Suresh 2013] investigates a finite element method working directly on tangled (*sic!*) meshes.

Elastic deformations. To recover local injectivity, we propose a method stemming from the computational physics. It is very important to note that there is rich literature on mesh deformation in the community working on grid generation for scientific computation. Numerical simulation of hydrodynamic instability of layered structures requires sound mathematical foundations behind moving deforming mesh algorithms. In the '60s Winslow and Crowley, independently one from another, introduced mesh generation methods based on inverse harmonic maps [Crowley 1962; Winslow 1966].

Since then, a lot of effort was spent on mesh generation based on elastic deformations [Jacquotte 1988], but mostly for regular grids. In 1988, at the time of domination of finite difference mapped grid generation methods, S. Ivanenko introduced the pioneering concept of barrier variational grid generations methods guaranteeing construction of non-degenerate grids [Charakhch'yan and Ivanenko 1997; Ivanenko 1988]. To generate deformations with bounded global distortion (bounded quasi-isometry constant), Garanzha proposed to minimize an elastic energy for a hyperelastic material with stiffening suppressing singular deformations [Garanzha 2000]. Invertibility theorem for deformation of this material was established in the 3D case as well [Garanzha et al. 2014].

A solid mathematical ground for these methods was laid by J. Ball who introduced in 1976 his theory of finite elasticity based on the concept of polyconvex distortion energies [Ball 1976]. He not only proved Weierstrass-style existence theorem for this class of variational problems, but also formulated a theorem on invertibility of elastic deformations for quite general 3D domains [Ball 1981]. It is important to note that Ball invertibility theorem is proved for Sobolev mappings and can be applied directly for finite element spaces, i.e. to deformation of meshes, as was pointed out in [Rumpf 1996].

Our contributions. Inspired by these results on untangling and elastic deformations, we propose a method outperforming recent state of the art on locally injective parameterization [Du et al. 2020] in terms of robustness, quality and supported features. To sum up, our contributions are:

- we propose a method that optimizes the map distortion directly, with a parameter for a tradeoff between the angle and area preservation,
- we provide a rigorous analysis of the foundations of our numerical resolution scheme,
- our method supports mapping with free boundaries,
- we observe better robustness for high distortions as well as with respect to a poor quality initialization.
- To ease the reproducibility, we publish the code in Python (refer to Listing 1) for a basic evaluation, and a C++ code in the supplemental material for more advanced applications.

The rest of the paper is organized as follows: we start with presenting our method in § 2, then we evaluate its performance (§ 3.1 and § 3.2) as well as its limitations (§ 3.3). Then we present theoretical guarantees for our resolution scheme: in § 4.1 we prove that our approximation of Hessian matrix is definite positive, and finally in § 4.2 we prove that our choice of the regularization parameter sequence guarantees that a minimization algorithm¹ can find a mesh free of inverted elements in a finite number of steps.

2 PENALTY METHOD FOR MESH UNTANGLING

In this rather short section we present our method of computing a foldover-free map $\vec{u} : \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}^d$, i.e. we map the domain Ω to a parametric domain. This presentation is unified both for 2D and 3D settings, and by d we denote the number of dimensions; in our notations we use arrows for all vectors of dimension d .

The section is organized as follows: in § 2.1 we give a primer on the variational formulation of mapping problem in continuous settings, then we state our problem in § 2.2 as a regularization of this variational formulation, and finally we present our numerical resolution scheme in § 2.3.

2.1 Variational formulation for grid generation

Let us denote by $\vec{u}(\vec{x})$ a map to a parametric domain: for the flat 2D case we can write as $\vec{u}(x, y) = (u(x, y), v(x, y))$, and for a 3D map $\vec{u}(x, y, z) = (u(x, y, z), v(x, y, z), w(x, y, z))$.

Consider the following variational problem:

$$\arg \min_{\vec{u}} \int_{\Omega} (f(J) + \lambda g(J)) \, dx, \quad (1)$$

where J is the Jacobian matrix of the mapping $\vec{u}(\vec{x})$, and

$$f(J) := \begin{cases} \frac{\text{tr} J^T J}{(\det J)^{\frac{d}{2}}}, & \det J > 0 \\ +\infty, & \det J \leq 0 \end{cases}$$

$$g(J) := \begin{cases} \det J + \frac{1}{\det J}, & \det J > 0 \\ +\infty, & \det J \leq 0 \end{cases}$$

Problem (1) may be subject to some constraints that we do not write explicitly. To give an example, one may pin some points in the map. In this formulation, functions $f(J)$ and $g(J)$ have concurrent goals, one preserves angles and the other preserves the area, and thus λ serves as a trade-off parameter.

¹The minimization algorithm is subject to conditions of Th. (1); from our numerical experiments we observe that our minimization algorithm almost always respects the conditions.

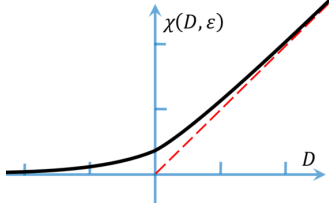


Fig. 2. Regularization function for the denominator in Eq. (3). When ε tends to zero, $\chi(\varepsilon, D)$ tends to D for positive values of D , and to 0^+ for negative values of D .

As a side note, with $\lambda = 0$ and $d = 2$, Prob. (1) presents a variational formulation of an inverse harmonic map problem. Namely, if we write down the Euler-Lagrange equations for Prob. (1) and interchange the dependent and independent variables², we obtain the Laplace equation $\Delta \vec{x}(\vec{u}) = \vec{0}$ (not to be confused with omnipresent $\Delta \vec{u}(\vec{x}) = \vec{0}$!). For this case, Prob. (1) is often referred to as Winslow's functional, however Winslow himself has never formulated the variational problem, working with inverse Laplace equations. To the best of our knowledge, the first publication of the variational problem is made by Brackbill and Saltzman [Brackbill and Saltzman 1982].

Prob. (1) provides a very powerful tool based on the theory of finite hyperelasticity introduced by J. Ball [Ball 1976]. From a computational point of view, starting with a map without foldovers, this tool allows us to optimize the quality of the map. Note that the energy (1) is a polyconvex function (refer to App. B–Rem. 3 for a proof) satisfying the ellipticity conditions, and therefore is very well suited for a numerical optimization provided that we have an initial guess in the admissible domain $\min_{\Omega} J(\vec{u}) > 0$.

The problem, however, is that while being theoretically sound, this problem statement does not offer any practical way to get rid of foldovers in a map, because for a map with foldovers the energy is infinite and provides no indications on how to improve the situation, hence we propose to alter a little the problem statement.

2.2 Penalty method

We can avoid nonpositive denominators in f and g using a regularization function χ for a positive value of ε (Fig. 2):

$$\chi(D, \varepsilon) := \frac{D + \sqrt{\varepsilon^2 + D^2}}{2}, \quad (2)$$

Then we define a regularized version $f_\varepsilon, g_\varepsilon$ of functions f and g :

$$f_\varepsilon(J) := \frac{\text{tr } J^T J}{(\chi(\det J, \varepsilon))^{\frac{2}{d}}}, \quad g_\varepsilon(J) := \frac{\det^2 J + 1}{\chi(\det J, \varepsilon)}, \quad (3)$$

so that Prob (1) is reformulated as

$$\lim_{\varepsilon \rightarrow 0^+} \arg \min_{\vec{u}} \int_{\Omega} (f_\varepsilon(J) + \lambda g_\varepsilon(J)) \, dx \quad (4)$$

²Attention, this step assumes that the solution of Prob. (1) is a diffeomorphism.

Under certain assumptions³ solutions of Prob. (4) are solutions of Prob. (1), however, Prob. (4) does offer a way of getting rid of foldovers if a foldover-free initialization is not available.

In practice, the map \vec{u} is piecewise affine with the Jacobian matrix J being piecewise constant, and can be represented by the coordinates of the vertices in the parametric domain $\{\vec{u}_i\}_{i=1}^{\#V}$. Let us denote the vector of all variables as $U := (\vec{u}_1^T \dots \vec{u}_{\#V}^T)^T$, then our optimization problem can be discretized as follows:

$$\lim_{\varepsilon \rightarrow 0^+} \arg \min_U F(U, \varepsilon), \quad (5)$$

$$\text{where } F(U, \varepsilon) := \sum_{t=1}^{\#T} (f_\varepsilon(J_t) + \lambda g_\varepsilon(J_t)) \text{vol}(T_t),$$

$\#V$ is the number of vertices, $\#T$ is the number of simplices, J_t is the Jacobian matrix for the simplex t and $\text{vol}(T_t)$ is the volume of the simplex T_t in the original domain.

2.3 Resolution scheme

To solve Prob. (5), we use an iterative descent method. We start from an initial guess U^0 , and we build a sequence of approximations $U^{k+1} := U^k + \Delta U^k$. For each iteration we need to carefully choose the regularization parameter ε^k . Starting from $\varepsilon^0 := 1$, we define the sequence as follows:

$$\varepsilon^{k+1} := \begin{cases} \left(1 - \frac{\sigma^k \sqrt{(D_-^{k+1})^2 + (\varepsilon^k)^2}}{|D_-^{k+1}| + \sqrt{(D_-^{k+1})^2 + (\varepsilon^k)^2}}\right) \varepsilon^k, & \text{if } D_-^{k+1} < 0, \\ (1 - \sigma^k) \varepsilon^k, & \text{if } D_-^{k+1} \geq 0, \end{cases} \quad (6)$$

where $D_-^{k+1} := \min_{t \in 1 \dots \#T} \det J_t^{k+1}$ is the minimum value of the Jacobian determinant over all cells of the mesh at the iteration $k + 1$, and $\sigma^k := \max\left(\frac{1}{10}, 1 - \frac{F(U^{k+1}, \varepsilon^k)}{F(U^k, \varepsilon^k)}\right)$. Note that Eq. (6) is justified by Th. 1 (§ 4.2) on finite untangling sequence; while this formula provides some guarantees, one may use a simpler heuristic to accelerate the computations, more on this in § 3.2.

The simplest way to find ΔU^k is to call a quasi-Newtonian solver such as L-BFGS-B [Zhu et al. 1997]. The only thing we need to implement is the computation of the function $F(U^k, \varepsilon^k)$ and its gradient $\nabla F(U^k, \varepsilon^k)$.

Another option is to compute analytically the Hessian matrix instead of estimating it. The problem, however, is that the Hessian matrix $\frac{\partial^2 F}{\partial U \partial U^T}$ is not positive definite. In this paper we propose its approximation that ensures the positive definiteness. The modified Hessian matrix $H^+(U^k, \varepsilon^k)$ of the function F with respect to U at the point U^k is built out of $d \times d$ blocks

$$H_{ij}^+ \approx \frac{\partial^2 F}{\partial \vec{u}_i \partial \vec{u}_j^T}(U^k, \varepsilon^k).$$

Here, the matrix H_{ij}^+ is placed on the intersection of i -th block row and j -th block column; the \approx symbol means that we remove all the terms depending on the second derivative of χ and second

³The fact that the solution of Prob. (4) is a diffeomorphism is sufficient (but not necessary) for the equivalence.

ALGORITHM 1: Computation of a locally injective map

Input: U^0 ; // initial guess (vector of size $\#V \times d$)
Input: *useQuasiNewton*; // boolean to choose the optimization scheme
Output: U ; // final locally injective map (vector of size $\#V \times d$)

```

1  $k \leftarrow 0$ ;
2 repeat
3   compute  $\varepsilon^k$ ; // regularization parameter, Eq. (6) [variant: Eq. (7)]
4   if useQuasiNewton then
5      $U^{k+1} \leftarrow \text{L-BFGS-B}(U^k, \varepsilon^k)$ ; // inner L-BFGS-B loop
6   else
7     compute a modified Hessian matrix  $H^+(U^k, \varepsilon^k)$ ;
8      $\Delta U^k \leftarrow (H^+)^{-1} \nabla F(U^k, \varepsilon^k)$ ; // conjugate gradients
9      $U^{k+1} \leftarrow \arg \min_{\tau} F(U^k + \tau \Delta U^k, \varepsilon^k)$ ; // line search
10  end
11   $k \leftarrow k + 1$ ;
12 until  $\min_{t \in \{1, \dots, T\}} \det J_t^k > 0$  and  $F(U^k, \varepsilon^k) > (1 - 10^{-3}) F(U^{k-1}, \varepsilon^{k-1})$ ;
13  $U \leftarrow U^k$ ;

```

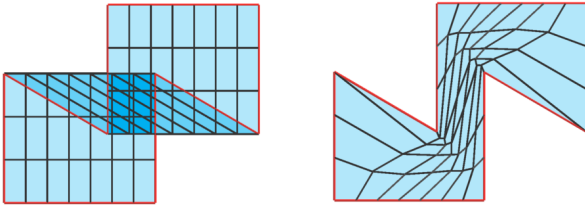


Fig. 3. The input mesh with foldovers and the untangling produced by the Listings 1 and 2. **Left:** a quad 2D mesh to untangle. The boundary (in red) is locked, and the black mesh is free to move. **Right:** fold-free result.

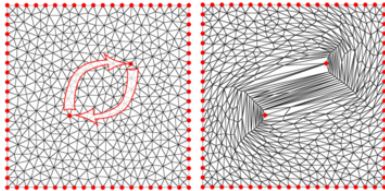


Fig. 4. Injective mapping sanity check: make two points inside a square switch places. Left: the input problem, all locked points are shown in red. Right: foldover-free result obtained with our method.

derivatives of $\det J$ to keep H^+ positive definite. Refer to Appendix A for the formulæ, and to § 4.1 for the proof of the positive definiteness of H^+ .

A detailed description of the resolution scheme is given in Alg. 1. Refer to List. 1 and Fig. 3 for a complete working example of Python implementation and the corresponding input/output generated by the code. Note that our method is not limited to simplicial meshes only: in this particular example we evaluate the Jacobian matrix for every triangle forming quad corners, what corresponds to the trapezoidal quadrature rule.

3 RESULTS AND DISCUSSION

In this section we provide an experimental evaluation of the method. In the field of computer graphics, any claim about map injectivity always faces a simple sanity check (Fig. 4): take a square and make two points inside switch places. Our method successfully avoids the desk-reject, so we start this section (§ 3.1) by testing our method on the benchmark [Du et al. 2020], then we continue with further tests we have found relevant (§ 3.2), and finally we discuss the limitations of the approach in § 3.3.

3.1 Benchmark database

Along with their paper, Du et al. have published a valuable benchmark database. It contains a huge number of 2D and 3D constrained boundary injective mapping challenges. For 2D challenges, the benchmark contains 3D triangulated surfaces to flatten, and not flat 2D meshes as we have described in § 2.1. Nevertheless, our method can handle it directly because the mapping is still $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ on each triangle. To the best of our knowledge, Total Lifted Content (TLC) [Du et al. 2020] and our method are the only ones passing the benchmark without any fail.

A representative example from the database is given in top row of Fig. 5. The challenge is to map the “Lucy” mesh statuette from the Stanford Computer Graphics Laboratory to a P-shaped domain. This mesh has a topology of a disk, and its boundary vertices are uniformly spaced on the P-shape boundary. As an initialization to the problem, Du et al. have computed the corresponding (flat) minimal surface that obviously contains a foldover (Fig. 5–top left). Then the problem boils down to a mesh untangling with locked boundary.

Mapping quality measure. How to measure quality of a map? Well, it depends on the goal. An identity is an unreachable ideal; traditional competing goals are (as much as possible) angle preserving and area preserving maps. Thus, we can measure the extreme values of the failure of a map to be conformal or aulhalic. Our maps being piecewise affine, the Jacobian matrix J is constant per element. Let us define the largest singular value of J as $\sigma_1(J)$, and the smallest singular value as $\sigma_d(J)$; then the quality of a mapping can be reduced to extreme values of the stretch ($\max \frac{\sigma_1(J)}{\sigma_d(J)}$) and the scaling ($\min \det J$).

For the “Lucy-to-P” challenge (Fig. 5–top row) our map differs from the TLC result by 12 orders of magnitude in terms of minimum scaling, and by two orders of magnitude in terms of maximum stretch. To visualize this difference in scaling, we have provided the close-ups: Fig. 5–top middle shows a map of the Lucy’s torch, whereas the same level of zoom on the result by Du et al. (Fig. 5–top right) contains not only the torch, but also both wings, the head and the right arm!

Note also that the input “Lucy” mesh is slightly anisotropic; our method allows us to prescribe the element target shape, so the dress pleats are clearly visible in our mapping.

Benchmark database. Our method successfully passes all challenges from the benchmark [Du et al. 2020]. The benchmark consists of 10743 meshes to untangle in 2D and 904 meshes in 3D under locked boundary constraints. In Fig. 7 we provide quality plots

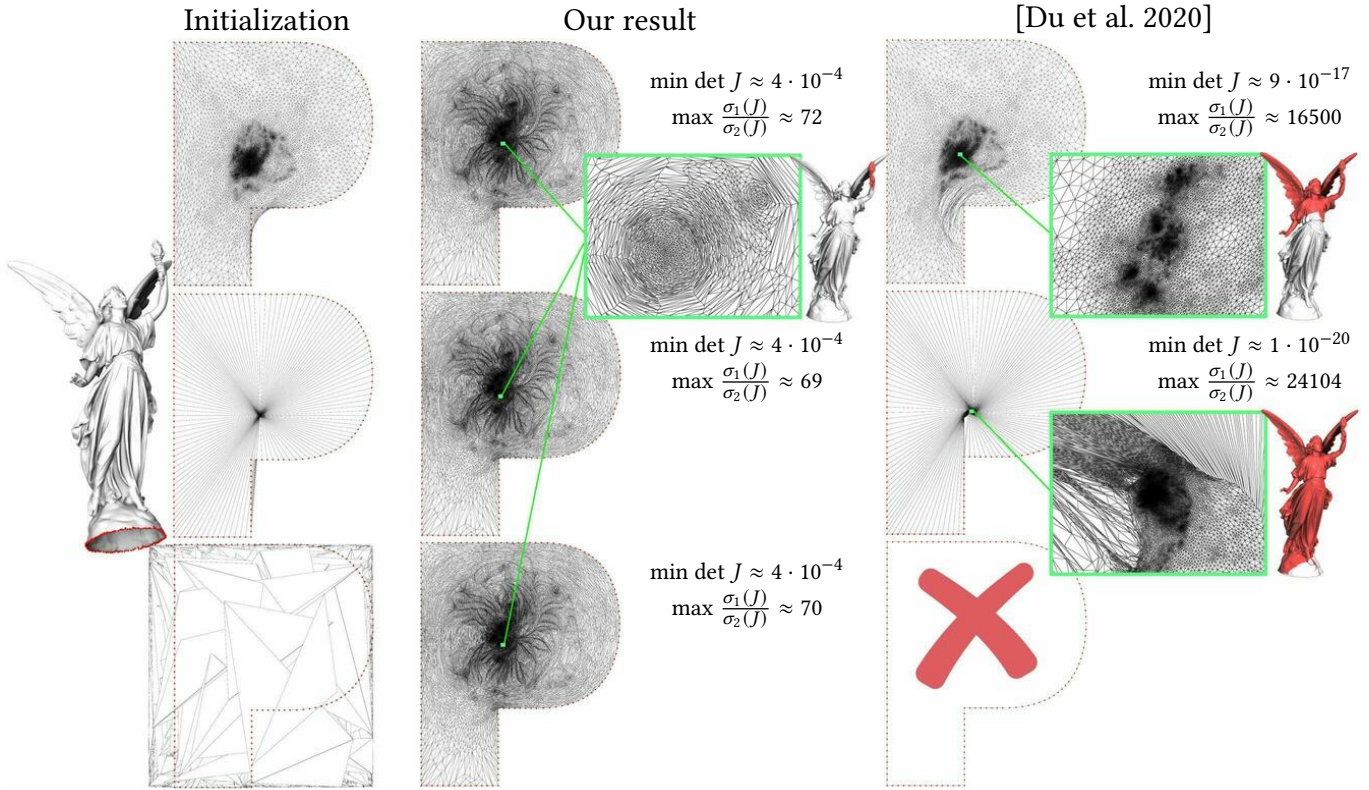


Fig. 5. Constrained boundary injective mapping challenge proposed by [Du et al. 2020]: the “Lucy” mesh is mapped to a P-shaped domain by constraining the vertices shown in red. Left column: three different initializations for the same problem. Middle column: our method produces the same (up to a numerical precision) result on all three initializations. Right column: total lifted content method [Du et al. 2020] fails to solve for the randomly initialized interior vertices, and produces very different results on other two initializations. Three miniature images of “Lucy” show in red the portion of the surface visible in the corresponding close-ups.

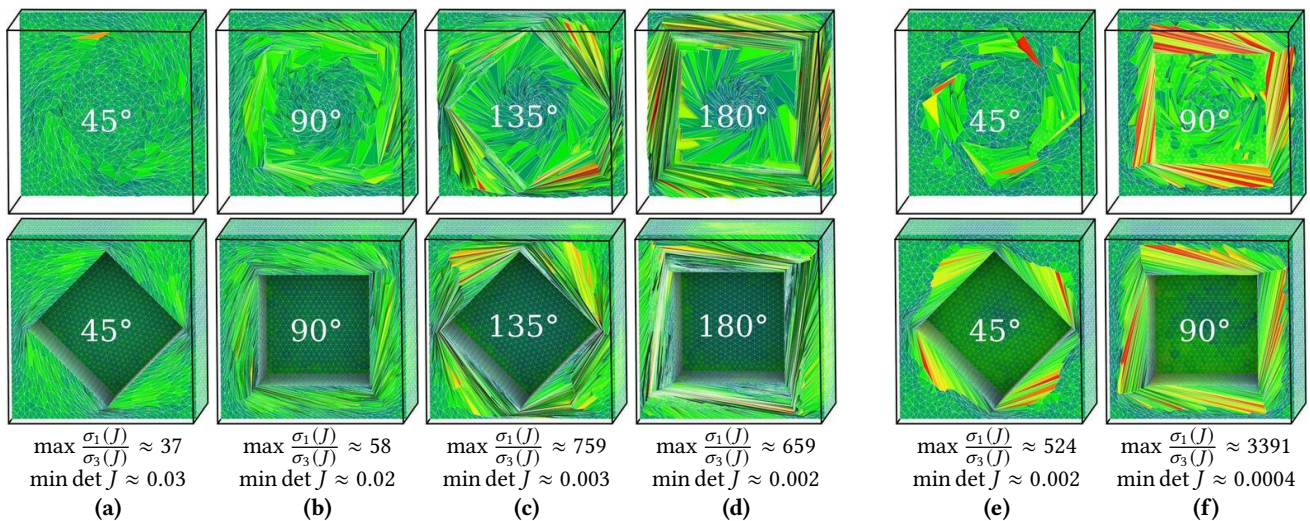


Fig. 6. Constrained boundary injective mapping stress test. We have generated an isotropic tetrahedral mesh of a cube subtracted from a larger cube, and we rotate the inner cube’s boundary to test the robustness. Top row and bottom row correspond to two different slices of the same mesh. Columns **(a)–(d)**: injective maps produced by our method, columns **(e)** and **(f)**: injective maps produced by [Du et al. 2020]. The method by [Du et al. 2020] fails to generate injective maps for 135° and 180° inner cube rotations. The colormap illustrates the relative volume scaling: green for $\det J \approx 1$, red for inflation, blue for compression.

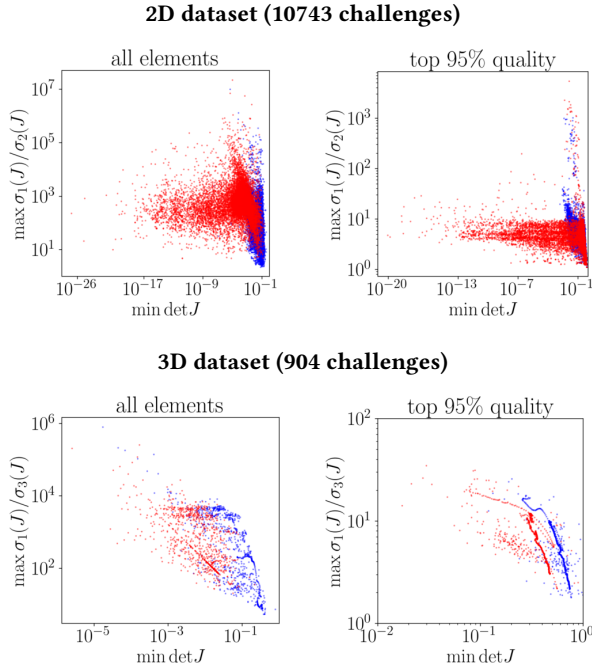


Fig. 7. Quality plot of the resulting locally injective maps for every challenge from the database provided by [Du et al. 2020]. Our results are shown in blue, whereas the results by Du et al. are shown in red. Each dot corresponds to a quality of the corresponding map reduced to two numbers: the maximum stretch and the minimum scale. **Top row**: mapping quality on the 2D dataset. **Bottom row**: mapping quality on the 3D dataset. Left column shows the absolute maximum stretch and absolute minimum scale, whereas the right column shows the maximum stretch and minimum scale for the top 95% of measurements.

of the resulting locally injective maps. These are log – log scatter plots: each dot corresponds to a quality of the corresponding map reduced to two numbers: the maximum stretch ($\max \frac{\sigma_1(J)}{\sigma_d(J)}$) and the minimum scaling ($\min \det J$). Left column of Fig. 7 shows the worst quality measurements for every 2D problem (top) as well as for every 3D challenge (bottom) of the dataset. Our results are shown in blue, whereas TLC results are shown in red. To illustrate the distribution of the elements’ quality, for each injective map we have removed 5% of worst measurements: the right column of Fig. 7 shows the maximum stretch and the minimum scaling for the top 95% of measurements.

Note the dot arrangements forming lines in the plot: these dots correspond to the few sequences of deformation present in the database.

Timings. Fig. 8 provides a log – log scatter plot of our running time vs mesh size for all the challenges from the database [Du et al. 2020]: for each run, the time varies from a fraction of a second to several minutes for the largest meshes. These times were obtained with a 12 cores i7-6800K CPU @ 3.40 GHz. As in Fig. 8, the vertical lines in the 3D dataset plot correspond to the sequences of deformation in the benchmark.

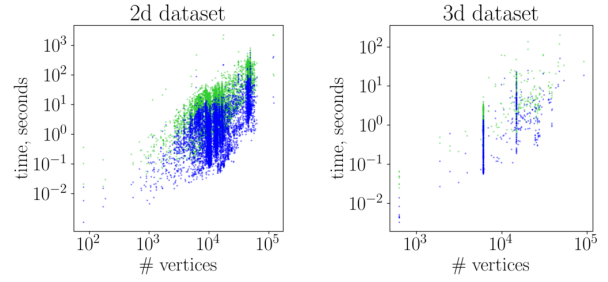


Fig. 8. Performance of our method tested on the benchmark [Du et al. 2020]. Each dot corresponds to a challenge from the database (10743 in 2D and 904 in 3D). Blue dots show the running times obtained using Eq. (7), green dots correspond to Eq. (6).

There are two scatter plots superposed, both represent the same resolution scheme with an exception corresponding to the way we compute ε^k (Alg. 1–line 3). The green scatter plot corresponds to a conservative update rule (Eq. (6)) offering guarantees on untangling in a finite number of steps (refer to Th. 1), whereas the blue scatter plot is obtained using the following update rule:

$$\varepsilon^k := \sqrt{10^{-12} + 4 \cdot 10^{-2} \cdot \left[\min(0, \min_{t \in 1 \dots \#T} \det J_t^k) \right]^2}, \quad (7)$$

This formula was chosen empirically, and it performs very well in the vast majority of situations. In particular, it allows for all the database [Du et al. 2020] to pass the injectivity test.

3.2 Further testing

Sensitivity to initialization. Our next test is the sensitivity to the initialization. We have generated two other initializations for the “Lucy-to-P” challenge: the one with all interior vertices collapsed onto a single point (Fig. 5–middle row), and with the interior vertices being randomly placed within a bounding square (Fig. 5–bottom row).

Our method produces virtually the same result on all three initializations, whereas TLC generates very different results for the first two, and fails for the third one. It is interesting to note that TLC is heavily depending on the initialization, it alters very little the input geometry.

Large deformation stress test. For our next test we have generated an isotropic tetrahedral mesh of a cube with a cavity, and we rotated the inner boundary to test the robustness of our method to large deformations. Figure 6 shows the results. Our L-BFGS-based optimization scheme succeeds up to the rotation of 135°, and we had to switch to the Newton method to reach the 180° rotation. TLC method had succeeded on 45° and 90°, and failed for the 135° and 180°. Note that as in the previous test, even when the untangling succeeds, TLC alters very little the input map, thus producing heavily stretched tetrahedra, whereas our method evenly dissipates the stress over all the domain.

Free boundary injective mapping. To the best of our knowledge, our method is the only one able to produce inversion-free maps

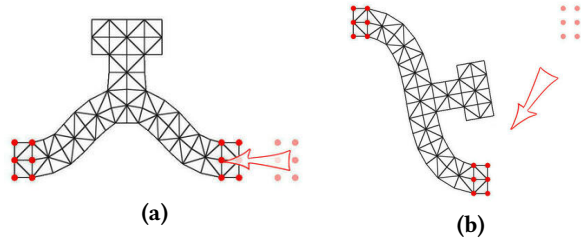


Fig. 9. Free boundary injective mapping. The vertices shown in red are constrained, all other vertices are free to move. (a): a compression test, (b): a bend test. Refer to Fig 10–a for the rest shape.

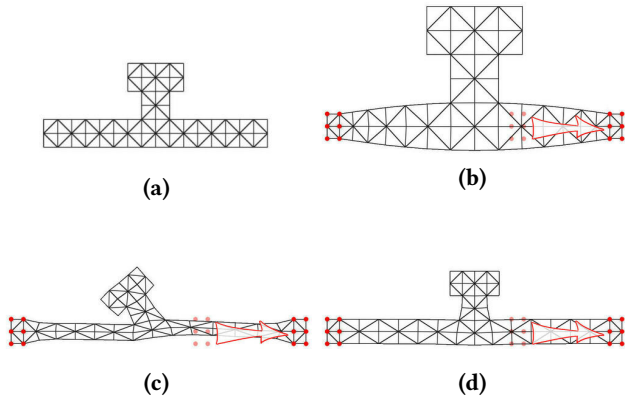


Fig. 10. Free boundary injective mapping; influence of the parameter λ in Prob. (5). The vertices shown in red are constrained, all other vertices are free to move. (a): the rest shape; (b): a stretch preserving the shape of the elements ($\lambda = 0$); (c): an area-preserving stretch ($\lambda = 10^4$); (d): a trade-off between the shape and the area preservation ($\lambda = 1$).

with free boundary. Since TLC tries to minimize the overall volume, relaxing the boundary constraints results in degenerate maps.

Fig. 9 shows two maps obtained with our method: a 2D shape being compressed and the same shape being bent. The boundary is free to move, we lock the vertices shown in red. Refer to Fig. 10–a for the rest shape. The shape behaves exactly as a human would expect it: upon compression the shape chooses one of the two possible results (Fig. 9–a), and successfully passes the bend test (Fig. 9–b), note the geometrical details that are naturally rotated.

Shape-area tradeoff λ . Our final tests illustrate the influence of the parameter λ in Prob. (5) on the resulting map. We have computed three free boundary maps of the rest shape (Fig. 10–a) being stretched. First we chose $\lambda = 0$, that is, only the shape quality term is taken into account in Prob. (5). When we optimize for the angles, the area of the triangles is forced to change, refer to Fig. 10–b for the resulting map. Naturally, an area preserving map ($\lambda = 10^4$) must deform the elements to satisfy the area constraint (Fig. 10–c). Finally, in Fig. 10–d we show an example with a tradeoff between the area and angles preservation.

3.3 Limitations

While globally performing very well in practice, our method still presents some limitations. We have two main sources of limitations: theoretical limitations as well as very practical ones related to numerical stability of our resolution scheme.

Overlaps. First of all, an inversion-free map does not imply global injectivity. Fig. 11–a provides an example of an inversion-free map with two cases of non-injectivity when optimizing for a map with free boundaries: the map can present global overlaps as well as the boundary can “wind up” around boundary vertices, i.e. the total angle of triangles incident to a vertex can be superior to 2π . Moreover, while being less frequent, similar situations may occur on interior vertices. Typically this situation happens near constraints causing a local compression in the shape.

Let us illustrate this behavior on a very simplistic mesh consisting of a single fan of 12 triangles. All vertices are free to move, the target shape is set to be the unit equilateral triangle for all elements. For this problem Fig. 12–a shows a local minimum, and the Fig. 12–c shows the global minimum respecting perfectly the prescribed total angle of 4π around the center vertex. Both are inversion-free maps, but only the map in Fig. 12–a is a globally injective one. Depending on the initialization and the resolution scheme chosen, we can converge to either minimum. Note, however, that the center vertex has the winding number 1 in one map and 2 in the other, and thus we can not deform continuously one to the other without inverting some elements. Note also that the configurations like in the Fig. 12–b present inverted elements and thus can not be generated by our method.

It is possible to avoid all overlaps altogether by embedding our shape to optimize into an outer triangulation, and performing a “bi-material” optimization. In this case, both global overlaps and fold-2-coverings are prohibited by the fact that the outer material must not have inverted elements (refer to Fig. 11–b). The thick prismatic layer in Fig. 1 was generated by a similar procedure: we have generated a very thin layer of triangular prisms around the dragon, and tetrahedralized the exterior bounded by a cube. After calling the untangling procedure, we have obtained an offset surface with exactly the same mesh connectivity as the original dragon mesh.

While this embedding kind of approach works well for certain applications, for other it may be hard to apply. We are currently exploring simpler practical ways to fix the winding problem.

Numerical challenges. Even when the problem is well-posed, a robust implementation may present significant difficulties. As we have said above, the quasi-Newtonian optimization scheme performs well for “simple” problems (it passes all the benchmark database!), but may fail for large deformations, where Newton iterations are necessary. While our modified Hessian matrix is symmetric positive definite, note that for stiff problems the Jacobi preconditioned conjugate gradients can fail and one might need the incomplete Choleski decomposition and beyond.

In practice we have found the method being very robust in 2D settings: we have not encountered a practical test case we were not able to treat with our method. In 3D, however, it can fail due

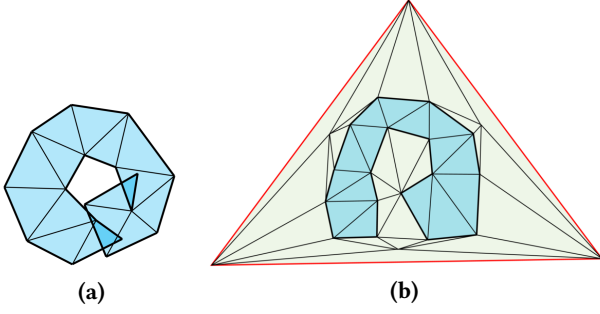


Fig. 11. Free boundary mapping limitations. **(a)**: this mesh presents two kinds of problems, namely, a global overlap and the mesh wrapped around a boundary vertex. **(b)**: Both problems can be avoided by embedding the mesh into an outer triangulation.

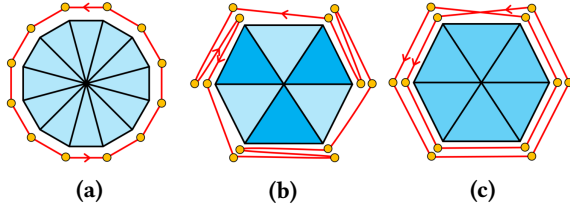


Fig. 12. Free boundary mapping limitations: three maps of a very simplistic mesh made of 12 triangles. **(a)** and **(c)** both are inversion-free maps and thus allowed by our method, whereas the map **(b)** has inverted elements, and thus is prohibited by our method.

to the numerical challenges in very anisotropic and highly twisted meshes.

4 ANALYSIS

This section presents in two parts a rigorous analysis of the penalty method. First in § 4.1 we prove that the modified Hessian matrix $H^+(U, \varepsilon)$ is indeed positive definite, and then in § 4.2 we show the origins of Eq. (6) for the regularization parameter sequence $\{\varepsilon^k\}_{k=0}^{\dots}$. Namely, we prove that if the problem has a solution, then an *idealized* minimization method can reach the admissible set $\min \det J > 0$ in a finite number of steps. An immediate consequence of this theorem is that if the problem has a solution, then for some $K < \infty$ the solution $\arg \min_U F(U, \varepsilon^K)$ belongs to the admissible set.

4.1 Modified Hessian matrix

Recall that in our resolution scheme we use the modified Hessian $(d \# V) \times (d \# V)$ matrix $H^+(U, \varepsilon)$ of the function $F(U, \varepsilon)$ built out of $d \times d$ blocks H_{ij}^+ placed on the intersection of i -th block row and j -th block column. It is a common practice to add some regularization terms to the Hessian matrix to make it positive definite, but we propose to modify the finite element (FE) matrix assembly procedure by eliminating some terms potentially leading to an indefinite FE matrix.

To this end, we restrict our attention to a single simplex and we study a function $\phi(J)$ of the Jacobian matrix defined as follows:

$$\phi(J) := f_\varepsilon(J) + \lambda g_\varepsilon(J) = \frac{\text{tr } J^\top J}{(\chi(\det J, \varepsilon))^{\frac{2}{d}}} + \lambda \frac{\det^2 J + 1}{\chi(\det J, \varepsilon)} \quad (8)$$

Let us denote by $a \in \mathbb{R}^{d^2}$ the (columnwise) flattening of the Jacobian matrix J , i.e. the vector composed of the elements of J . We decompose the $d^2 \times d^2$ Hessian matrix of ϕ with respect to the Jacobian matrix entries into two parts: $\frac{\partial^2 \phi}{\partial a \partial a^\top} = M^+ + M^\pm$, where M^+ is a positive definite matrix, and M^\pm can be an indefinite matrix that we neglect. The matrix M^\pm contains all terms depending on χ'' and second derivatives of $\det J$ with respect to elements of the Jacobian matrix J . Our map is affine on the simplex of interest, therefore its Jacobian matrix J is a linear function of the vertices of the simplex. Thus, the idea is to compute a positive definite matrix $M^+(J)$, and use the chain rule to get the Hessian matrix with respect to our variables U and assemble the matrix H^+ .

So, we choose some arbitrary point J_0 and we want to show the way to decompose $\frac{\partial^2 \phi}{\partial a \partial a^\top}(J_0)$ into a sum of $M^+(J_0)$ and $M^\pm(J_0)$ with $M^+(J_0) > 0$. To do so, first we write down the first order Taylor expansion $q(D)$ of the function $\chi(D, \varepsilon)$ around some point $D_0 = \det J_0$:

$$q(D) := \chi(D_0, \varepsilon) + \frac{\partial \chi}{\partial D}(D_0, \varepsilon)(D - D_0).$$

Next we define a function $\Phi(a, D)$ as follows:

$$\Phi(a, D) := \frac{|a|^2}{(q(D))^{\frac{2}{d}}} + \lambda \frac{D^2 + 1}{q(D)}.$$

Note that Φ differs a bit from ϕ : it has one more argument and χ is replaced by its linearization in the denominator. While this maneuver might seem obscure, light will be shed very shortly. Φ has a major virtue of being convex! The convexity is easy to prove, refer to Appendix B for a formal proof.

Having built a convex function Φ , it is straightforward to verify that the following decomposition holds:

$$\frac{\partial^2 \phi}{\partial a \partial a^\top}(J_0) = M^+(J_0) + M^\pm(J_0), \quad (9)$$

where

$$M^+ := \left(I \quad \frac{\partial D}{\partial a} \right) \begin{pmatrix} \frac{\partial^2 \Phi}{\partial a \partial a^\top} & \frac{\partial^2 \Phi}{\partial a \partial D} \\ \frac{\partial^2 \Phi}{\partial D \partial a^\top} & \frac{\partial^2 \Phi}{\partial D^2} \end{pmatrix} \begin{pmatrix} I & \frac{\partial D}{\partial a} \end{pmatrix}^\top, \text{ and}$$

$$M^\pm := \frac{\partial \Phi}{\partial D} \frac{\partial^2 D}{\partial a \partial a^\top} - \frac{\chi''}{\chi} \left(\frac{2}{d} f_\varepsilon + \lambda g_\varepsilon \right) \frac{\partial D}{\partial a} \frac{\partial D}{\partial a^\top}.$$

The easiest way to check that the equality (9) holds is to note that at the point J_0 we have $q = \chi$, $q' = \chi'$, and therefore we have

$$\frac{\partial \phi(a)}{\partial a} = \frac{\partial \Phi(a, D(a))}{\partial a} + \frac{\partial \Phi(a, D(a))}{\partial D} \frac{\partial D(a)}{\partial a}.$$

To calculate the Hessian $\frac{\partial^2 \phi}{\partial a \partial a^\top}(J_0)$, it suffices to differentiate this expression one more time and add the terms in χ'' that were zeroed out by the linearization.

To sum up, in our computations, for each simplex we approximate the Hessian matrix $\frac{\partial^2 \phi}{\partial a \partial a^\top}$ by the $d^2 \times d^2$ matrix M^+ and we neglect the term M^\pm . Thanks to the convexity of Φ , it is trivial to verify that

for any choice of J_0 the matrix M^+ is definite positive. Then we use the chain rule over M^+ to get the Hessian matrix with respect to our variables U , and we assemble a $(d \#V) \times (d \#V)$ approximation H^+ of the Hessian matrix for the energy function $F(U, \varepsilon)$. Matrix H^+ is positive definite provided that at least d mesh vertices are fixed. If less than d points are fixed, rigid body transformations are allowed. The energy is invariant w.r.t rigid body transformations, so when constraints allow for such transformations, matrix H^+ becomes positive semidefinite. Note that the leading blocks H_{ii}^+ are always positive definite. Refer to Appendix A for further details on the finite element assembly procedure.

4.2 Choice of ε^k

In this section we provide a strategy for the choice of the regularization parameter ε^k at each iteration. Namely, we prove that an idealized minimization algorithm reaches the admissible set $\min \det J > 0$ in a finite number of iterations.

THEOREM 1. *Let us suppose that the admissible set is not empty, namely there exists a mesh U^* satisfying $F(U^*, 0) < +\infty$. We also suppose that we have a minimization algorithm satisfying one of the following efficiency conditions for some $0 < \sigma < 1$:*

- **Either** the essential descent condition holds

$$F(U^{k+1}, \varepsilon^k) \leq (1 - \sigma)F(U^k, \varepsilon^k), \quad (10)$$

- **or** the vector U^k satisfies the quasi-minimality condition:

$$\min_U F(U, \varepsilon^k) > (1 - \sigma)F(U^k, \varepsilon^k). \quad (11)$$

Then the admissible set is reachable by solving a finite number of minimization problems in U with ε^k fixed for each problem. In other words, under a proper choice of the regularization parameter sequence $\varepsilon^k, k = 0 \dots K$, we obtain $F(U^K, 0) < +\infty$.

PROOF. The main idea is to expose an explicit way to build a decreasing sequence $\{\varepsilon^k\}_{k=0}^\infty$ such that the sequence $\{F(U^k, \varepsilon^k)\}_{k=0}^\infty$ is bounded from above. Then we can prove that the admissible set is reachable in a finite number of steps by a simple *reductio ad absurdum* argument.

First of all, the function $F(U, \varepsilon)$ can be rewritten as follows

$$F(U, \varepsilon) = \sum_i \alpha_i \frac{\psi_i(U)}{\chi(D_i, \varepsilon)}, \quad (12)$$

where $D_i = D_i(U)$ denotes the Jacobian determinant for i -th simplex of the mesh ($D_i = \det J_i$), and $\alpha_i > 0$ are positive, separated from zero weights assigned to each simplex. The functions

$$\psi_i(U, \varepsilon) := \chi(D_i, \varepsilon)^{1-\frac{2}{d}} \operatorname{tr} J_i^T J_i + \lambda(D_i^2 + 1)$$

defined according to Eq.(3) are positive and bounded from below as

$$\alpha_i \psi_i(U, \varepsilon) \geq \lambda \min_i \alpha_i.$$

Note also that $\psi_i(U, \varepsilon)$ are increasing functions of ε .

Our goal is to build a decreasing sequence $\{\varepsilon^k\}_{k=0}^\infty$ such that the sequence $\{F(U^k, \varepsilon^k)\}_{k=0}^\infty$ is bounded from above. We split the construction into two parts: first we suppose that at some iteration k the essential condition (10) is satisfied, and then we explore the case (11).

Suppose that the condition (10) holds at iteration k . In order to guarantee that the function does not increase, it suffices to establish the following inequality:

$$(1 - \sigma)F(U^{k+1}, \varepsilon^{k+1}) \leq F(U^{k+1}, \varepsilon^k). \quad (13)$$

By noting that $\psi_i(U^{k+1}, \varepsilon^{k+1}) \leq \psi_i(U^{k+1}, \varepsilon^k)$, Ineq. (13) is implied if the following condition holds:

$$\forall i: \quad (1 - \sigma)\chi(D_i^{k+1}, \varepsilon^k) \leq \chi(D_i^{k+1}, \varepsilon^{k+1}) \quad (14)$$

where $D_i^{k+1} := D_i(U^{k+1})$ denotes the Jacobian determinant of simplex i at iteration $k + 1$.

Let us show a constructive way to build ε^{k+1} such that Ineq. (14) is satisfied. To do so, first note that χ is convex with respect to the second argument, hence

$$\chi(D_i^{k+1}, \varepsilon^{k+1}) - \chi(D_i^{k+1}, \varepsilon^k) \geq (\varepsilon^{k+1} - \varepsilon^k) \frac{\partial}{\partial \varepsilon} \chi(D_i^{k+1}, \varepsilon^k).$$

Thus, since χ and $\frac{\partial}{\partial \varepsilon} \chi(D, \varepsilon)$ are both positive, Ineq. (14) holds provided that

$$\forall i: \quad \varepsilon^{k+1} \geq \varepsilon^k - \sigma \frac{\chi(D_i^{k+1}, \varepsilon^k)}{\frac{\partial}{\partial \varepsilon} \chi(D_i^{k+1}, \varepsilon^k)} \quad (15)$$

In its turn, since $\frac{\chi(D, \varepsilon)}{\frac{\partial}{\partial \varepsilon} \chi(D, \varepsilon)}$ is an increasing function of D , condition (15) can be simplified to

$$\varepsilon^{k+1} \geq \varepsilon^k - \sigma \frac{\chi(D_-^{k+1}, \varepsilon^k)}{\frac{\partial}{\partial \varepsilon} \chi(D_-^{k+1}, \varepsilon^k)},$$

where $D_-^{k+1} := \min_i D_i^{k+1}$ is the minimum value of the Jacobian determinant over all simplices.

Hence, if U^{k+1} is an approximate solution of the minimization problem $\arg \min_U F(U, \varepsilon^k)$ with fixed parameter ε^k , we can use the following update rule for ε^{k+1} :

$$\varepsilon^{k+1} = \begin{cases} \varepsilon^k - \sigma \frac{\chi(D_-^{k+1}, \varepsilon^k)}{\frac{\partial}{\partial \varepsilon} \chi(D_-^{k+1}, \varepsilon^k)}, & \text{if } D_-^{k+1} < 0, \\ \varepsilon^k - \sigma \frac{\chi(0, \varepsilon^k)}{\frac{\partial}{\partial \varepsilon} \chi(0, \varepsilon^k)}, & \text{if } D_-^{k+1} \geq 0, \end{cases} \quad (16)$$

This update rule guarantees that Ineq. (13) is satisfied; coupled with the assumption (10) of the theorem, this implies the required non-growth property of the function values sequence:

$$F(U^{k+1}, \varepsilon^{k+1}) \leq F(U^k, \varepsilon^k). \quad (17)$$

Consider now the case where condition (11) holds at iteration k . Note that condition (11) essentially means that our current solution U^k is very close to the global minimum of $F(U, \varepsilon^k)$, and thus Ineq. (10) cannot be satisfied. Nevertheless, we can use the same update rule (16) for computation of ε^{k+1} . Indeed, with this choice we have

$$\begin{aligned} F(U^{k+1}, \varepsilon^{k+1}) &\leq \frac{1}{(1 - \sigma)} F(U^{k+1}, \varepsilon^k) \leq \frac{1}{(1 - \sigma)} F(U^k, \varepsilon^k) < \\ &< \frac{1}{(1 - \sigma)^2} \min_U F(U, \varepsilon^k) < \frac{1}{(1 - \sigma)^2} \min_U F(U, 0). \end{aligned}$$

Here the last inequality provides a global bound on the function values sequence, and it is based on the observation $\frac{\partial}{\partial \varepsilon} \chi(D, \varepsilon) > 0$.

To sum up, we have shown a way to build a sequence $\{\varepsilon^k\}_{k=0}^\infty$ such that the sequence $\{F(U^k, \varepsilon^k)\}_{k=0}^\infty$ is bounded from above. Now let us prove that the update rule (16) allows to reach the admissible set in a finite number of steps. To do so, we use a *reductio ad absurdum* argument.

Suppose that the admissible set is never reached for an infinite decreasing sequence $\{\varepsilon^k\}_{k=0}^\infty$ built using the update rule (16), i.e. $D_-^{k+1} < 0 \forall k \geq 0$.

Then, using the facts that $\frac{\partial}{\partial \varepsilon} \chi(D, \varepsilon) \leq \frac{1}{2}$ and $\{\varepsilon^k\}_{k=0}^\infty$ form a decreasing sequence of non-negative values, we can rewrite the update rule (16) for some $K > 0$:

$$\varepsilon^0 - \varepsilon^K \geq 2\sigma \sum_{k=0}^{K-1} \chi(D_-^{k+1}, \varepsilon^k) \geq 2\sigma K \min_{0 \leq k < K} \chi(D_-^{k+1}, \varepsilon^k),$$

with an immediate consequence that for an arbitrarily large K we have the following inequality:

$$\max_{0 \leq k < K} \frac{1}{\chi(D_-^{k+1}, \varepsilon^k)} \geq \frac{2\sigma K}{\varepsilon^0}.$$

Since all terms $\alpha_i \psi_i(U)$ in (12) are bounded from below, the resulting estimate contradicts the boundedness of $F(U^k, \varepsilon^k)$, thus concluding our proof. \square

REMARK 1. *An important corollary of Th. 1 is that, provided that the admissible set is not empty, there exists an iteration $K < \infty$ such that the global minimum of the function $F(U, \varepsilon^K)$ belongs to the admissible set. The proof is rather obvious: suppose we have an idealized minimizer such that $U^{k+1} = \arg \min_U F(U, \varepsilon^k)$. This minimizer always*

satisfies the conditions of Th. 1, therefore it can untangle the mesh in a finite number of steps.

REMARK 2. *For our choice of the regularization function χ , the update rule (16) can be instantiated as follows:*

$$\varepsilon^{k+1} = \begin{cases} \left(1 - \frac{\sigma \sqrt{(D_-^{k+1})^2 + (\varepsilon^k)^2}}{|D_-^{k+1}| + \sqrt{(D_-^{k+1})^2 + (\varepsilon^k)^2}}\right) \varepsilon^k, & \text{if } D_-^{k+1} < 0, \\ (1 - \sigma) \varepsilon^k, & \text{if } D_-^{k+1} \geq 0, \end{cases} \quad (18)$$

In practice the global estimate σ is not known in advance, and the optimization routine may be far from the ideal. For each minimization step we compute the local descent coefficient σ^k :

$$\sigma^k := 1 - \frac{F(U^{k+1}, \varepsilon^k)}{F(U^k, \varepsilon^k)}.$$

When $\sigma^k \geq \sigma$ one can use the update rule (18) using the local value σ^k guaranteeing that Ineq. (17) holds. In the case $\sigma^k < \sigma$ one should check that condition (11) holds for prescribed σ . If positive, we can assign $\sigma^k = \sigma$ and use update rule (18). If one cannot assure (11), it means that minimization procedure for ε^k failed and theorem cannot be applied (it does not mean that Alg. 1 will not reach the admissible set!). In numerical experiments we use value $\sigma = \frac{1}{10}$.

5 CONCLUSION

Producing maps without reverted elements is a major challenge in geometry processing. Inspired by untangling solutions in computational physics, our solution outperforms the state of the art both in terms of robustness and distortion. It is easy to use since we provide a simple implementation that is free of commercial product dependency (compiler, library, etc.). Moreover, the energy is estimated independently on each triangle / tetrahedra, making it a good candidate to be adapted to more difficult settings including free boundary and global parameterization.

REFERENCES

- Noam Aigerman and Yaron Lipman. 2013. Injective and Bounded Distortion Mappings in 3D. *ACM Trans. Graph.* 32, 4, Article 106 (July 2013), 14 pages. <https://doi.org/10.1145/2461912.2461931>
- John M Ball. 1976. Convexity conditions and existence theorems in nonlinear elasticity. *Archive for rational mechanics and Analysis* 63, 4 (1976), 337–403.
- J. M. Ball. 1981. Global invertibility of Sobolev functions and the interpenetration of matter. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics* 88, 3-4 (1981), 315–328. <https://doi.org/10.1017/S030821050002014X>
- 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. <https://doi.org/10.1145/2461912.2462014>
- J.U Brackbill and J.S Saltzman. 1982. Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys.* 46, 3 (1982), 342 – 368. [https://doi.org/10.1016/0021-9991\(82\)90020-1](https://doi.org/10.1016/0021-9991(82)90020-1)
- 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. <https://doi.org/10.1145/2897824.2925890>
- AA Charakhch'yan and SA Ivanenko. 1997. A variational form of the Winslow grid generator. *J. Comput. Phys.* 136, 2 (1997), 385–398.
- WP Crowley. 1962. An equipotential zoner on a quadrilateral mesh. *Memo, Lawrence Livermore National Lab* 5 (1962).
- Josh Danczyk and Krishnan Suresh. 2013. Finite element analysis over tangled simplicial meshes: Theory and implementation. *Finite Elements in Analysis and Design* 70-71 (2013), 57 – 67. <https://doi.org/10.1016/j.finel.2013.04.004>
- Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z. Kovalsky, Yajie Yan, Danny M. Kaufman, and Tao Ju. 2020. Lifting Simplices to Find Injectivity. *ACM Trans. Graph.* 39, 4, Article 120 (July 2020), 17 pages. <https://doi.org/10.1145/3386569.3392484>
- Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbury, and Werner Stuetzle. 1995. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 173–182. <https://doi.org/10.1145/218380.218440>
- José Maria Escobar, Eduardo Rodriguez, Rafael Montenegro, Gustavo Montero, and José Maria González-Yuste. 2003. Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering* 192, 25 (2003), 2775–2787.
- Michael S. Floater. 1997. Parametrization and Smooth Approximation of Surface Triangulations. *Comput. Aided Geom. Des.* 14, 3 (April 1997), 231–250. [https://doi.org/10.1016/S0167-8396\(96\)00031-3](https://doi.org/10.1016/S0167-8396(96)00031-3)
- Lori A Freitag and Paul Plassmann. 2000. Local optimization-based simplicial mesh untangling and improvement. *Internat. J. Numer. Methods Engrg.* 49, 1-2 (2000), 109–125.
- VA Garanzha. 2000. The barrier method for constructing quasi-isometric grids. *Computational Mathematics and Mathematical Physics* 40 (2000), 1617–1637.
- V.A. Garanzha, L.N. Kudryavtseva, and S.V. Utyuzhnikov. 2014. Variational method for untangling and optimization of spatial meshes. *J. Comput. Appl. Math.* 269 (2014), 24 – 41. <https://doi.org/10.1016/j.cam.2014.03.006>
- J. Gregson, A. Sheffer, and E. Zhang. 2011. All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Computer Graphics Forum (Special Issue of Symposium on Geometry Processing 2011)* 30, 5 (2011).
- Kai Hormann, Konrad Polthier, and Alia Sheffer. 2008. Mesh Parameterization: Theory and Practice. In *ACM SIGGRAPH ASIA 2008 Courses* (Singapore) (SIGGRAPH Asia '08). Association for Computing Machinery, New York, NY, USA, Article 12, 87 pages. <https://doi.org/10.1145/1508044.1508091>
- SA Ivanenko. 1988. Construction of nondegenerate grids. *Zh. Vychisl. Mat. Mat. Fiz* 28, 10 (1988), 1498.
- Olivier-P Jacquotte. 1988. A mechanical model for a new grid generation method in computational fluid dynamics. *Computer methods in applied mechanics and engineering* 66, 3 (1988), 323–338.

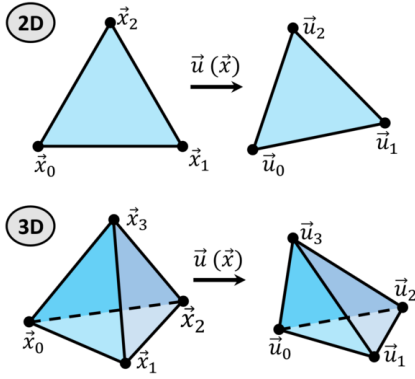


Fig. 13. On each simplex the map $\vec{u}(\vec{x})$ is affine and is entirely defined by the position of the vertices of the domain simplex $\{\vec{x}_i\}$ and its image $\{\vec{u}_i\}$.

- Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph.* 36, 6, Article 186 (Nov. 2017), 9 pages. <https://doi.org/10.1145/3130800.3130895>
- Zhongshi Jiang, Teseo Schneider, Denis Zorin, and Daniele Panozzo. 2020. Bijective Projection in a Shell. *ACM Trans. Graph.* 39, 6, Article 247 (Nov. 2020), 18 pages. <https://doi.org/10.1145/3414685.3417769>
- Patrick M Knupp. 2001. Hexahedral and tetrahedral mesh untangling. *Engineering with Computers* 17, 3 (2001), 261–268.
- Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection-and Inversion-Free, Large-Deformation Dynamics. *ACM Trans. Graph.* 39, 4, Article 49 (July 2020), 20 pages. <https://doi.org/10.1145/3386569.3392425>
- Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Trans. Graph.* 31, 4, Article 108 (July 2012), 13 pages. <https://doi.org/10.1145/2185520.2185604>
- Matthias Nieser, Ulrich Reitebuch, and Konrad Polthier. 2011. CubeCover - Parameterization of 3D Volumes. *Computer Graphics Forum* (2011). <https://doi.org/10.1111/j.1467-8659.2011.02014.x>
- Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2, Article 16 (April 2017), 16 pages. <https://doi.org/10.1145/2983621>
- Martin Rumpf. 1996. A variational approach to optimal meshes. *Numer. Math.* 72, 4 (1996), 523–540.
- Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum (proceedings of Symposium on Geometry Processing)* 32, 5 (2013).
- Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. 2019. Progressive Embedding. *ACM Trans. Graph.* 38, 4, Article 32 (July 2019), 13 pages. <https://doi.org/10.1145/3306346.3323012>
- Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (July 2015), 9 pages. <https://doi.org/10.1145/2766947>
- Thomas Toulorge, Christophe Geuzaine, Jean-François Remacle, and Jonathan Lambrechts. 2013. Robust untangling of curvilinear meshes. *J. Comput. Phys.* 254 (2013), 8–26.
- W. T. Tutte. 1963. How to Draw a Graph. *Proceedings of the London Mathematical Society* s3-13, 1 (01 1963), 743–767. <https://doi.org/10.1112/plms/s3-13.1.743> arXiv:[https://academic.oup.com/plms/article-pdf/s3-13/1/743/4385170/s3-13-1-743.pdf](https://arxiv.org/abs/https://academic.oup.com/plms/article-pdf/s3-13/1/743/4385170/s3-13-1-743.pdf)
- Alan M Winslow. 1966. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. *Journal of computational physics* 1, 2 (1966), 149–172.
- C. Zhu, R. H. Byrd, and J. Nocedal. 1997. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Trans. Math. Software* 23, 4 (1997), 550–560.

A COMPREHENSIVE DESIGN FORMULÆ

Given a map \vec{u} , let us denote by \vec{a}_i , $i = 1, 2, (, 3)$ the tangent basis, i.e. vectors forming the columns of the Jacobian matrix J . For example, in 2D we have $\vec{a}_1 := \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \end{pmatrix}^\top$ and $\vec{a}_2 := \begin{pmatrix} \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{pmatrix}^\top$. Let us denote by \vec{b}_i the dual basis, i.e. vectors chosen in the way that $\vec{a}_i^\top \vec{b}_j = \delta_{ij} \det J$ for all indices i, j . In particular, for the 2D settings the dual basis can be written as $\vec{b}_1 := \begin{pmatrix} \frac{\partial v}{\partial y} & -\frac{\partial u}{\partial y} \end{pmatrix}^\top$ and $\vec{b}_2 := \begin{pmatrix} -\frac{\partial v}{\partial x} & \frac{\partial u}{\partial x} \end{pmatrix}^\top$. In the 3D case $\vec{b}_k = \vec{a}_i \times \vec{a}_j$, where i, j, k is cyclic permutation from 1, 2, 3. It is a handy choice of variables, in particular, $\text{tr} J^\top J = \sum_i |\vec{a}_i|^2$ and $\frac{\partial \det J}{\partial \vec{a}_i} = \vec{b}_i$. For further simplification of notations we will use χ for $\chi(D, \varepsilon)$, χ' for $\frac{\partial \chi(D, \varepsilon)}{\partial D}$ and $a^\top = (\vec{a}_1^\top \dots \vec{a}_d^\top)$, $b^\top = (\vec{b}_1^\top \dots \vec{b}_d^\top)$.

A.1 Gradient

In order to derive expressions for the gradient and the Hessian matrix of F , we write down explicitly the Jacobian matrix J for the affine map of a simplex T with vertices $\vec{u}_0, \vec{u}_1, \dots, \vec{u}_d$:

$$J = (\vec{a}_1 \dots \vec{a}_d) = (\vec{u}_1 - \vec{u}_0 \ \vec{u}_2 - \vec{u}_0 \ \dots \ \vec{u}_d - \vec{u}_0) S^{-1} = (\vec{u}_0 \dots \vec{u}_d) Z,$$

where

$$S := (\vec{x}_1 - \vec{x}_0 \ \vec{x}_2 - \vec{x}_0 \ \dots \ \vec{x}_d - \vec{x}_0), \quad \det S > 0$$

is the shape matrix, \vec{x}_i are vertices of “ideal” or “target” shape for the image of the simplex T , and Z is a $(d+1) \times d$ matrix defined as

$$Z := \{z_{ij}\} := \begin{pmatrix} -1 & \dots & -1 \\ & I & \end{pmatrix} S^{-1}$$

Since the Jacobian matrix is a linear function of \vec{u}_i , we have

$$\frac{\partial \vec{a}_i}{\partial \vec{u}_j} = z_{ji} I, \quad i = 1, \dots, d, \quad j = 0, \dots, d.$$

The additive contribution to gradient of F from the simplex T can be written using correspondence of local indices 0 – d and global indices $g_0 - g_d$ in the list of vertices:

$$\begin{aligned} (\nabla F)_{g_j} &+ = \frac{\det S}{d!} \sum_{i=1}^d \frac{\partial \vec{a}_i^\top}{\partial \vec{u}_j} \frac{\partial \phi}{\partial \vec{a}_i} = \\ &= \frac{\det S}{d!} \sum_{i=1}^d z_{ji} \frac{\partial \phi}{\partial \vec{a}_i}, \quad j = 0, \dots, d, \end{aligned}$$

where function $\phi(J)$ is defined in § 4.1. Let us provide an explicit expression for $\frac{\partial \phi}{\partial \vec{a}_i}$:

$$\frac{\partial \phi}{\partial \vec{a}_i} = \frac{2}{\chi^{\frac{1}{2}}} \vec{a}_i - \frac{1}{\chi} \left(\frac{2}{d} f_\varepsilon \chi' - 2\lambda \det J + \lambda g_\varepsilon \chi' \right) \vec{b}_i$$

A.2 Hessian

The blocks of the nonnegative definite part of Hessian matrix of F can be updated using the following general formula

$$H_{g_j g_i}^+ + = \frac{\det S}{d!} \sum_{m,l} \frac{\partial \vec{a}_m^\top}{\partial \vec{u}_j} M_{ml}^+ \frac{\partial \vec{a}_l}{\partial \vec{u}_i},$$

where M_{ml}^+ denotes a $d \times d$ block of $d^2 \times d^2$ positive definite matrix M^+ defined in Eq. (9). Let us provide an explicit expression for the matrix:

$$M^+ = (I \quad b) \begin{pmatrix} \frac{\partial^2 \Phi}{\partial a \partial a^\top} & \frac{\partial^2 \Phi}{\partial a \partial D} \\ \frac{\partial^2 \Phi}{\partial D \partial a^\top} & \frac{\partial^2 \Phi}{\partial D^2} \end{pmatrix} (I \quad b)^\top, \text{ where}$$

$$\frac{\partial^2 \Phi}{\partial a \partial a^\top} = \frac{2}{\chi^{\frac{2}{d}}} I$$

$$\frac{\partial^2 \Phi}{\partial D^2} = \frac{2}{d} \left(1 + \frac{2}{d}\right) |a|^2 \frac{\chi'^2}{\chi^{2+\frac{2}{d}}} + \lambda \left(\frac{2}{\chi} - 4D \frac{\chi'}{\chi^2} + 2(1+D^2) \frac{\chi'^2}{\chi^3} \right)$$

$$\frac{\partial^2 \Phi}{\partial a \partial D} = -\frac{4}{d} \frac{\chi'}{\chi^{1+\frac{2}{d}}} a.$$

Obviously the leading $d \times d$ blocks of the matrix H^+ are strictly positive definite and can be used to build Newton-type minimization algorithm.

B CONVEXITY OF Φ

Recall that the function Φ is defined as follows (§ 4.1):

$$\Phi(a, D) := \frac{|a|^2}{q^{\frac{2}{d}}} + \lambda \frac{D^2 + 1}{q},$$

where $a \in \mathbb{R}^{d^2}$ is the (columnwise) flattening of the Jacobian matrix J , i.e. the vector composed of the elements of J .

LEMMA 1. $\nabla \nabla^\top \Phi > 0$

PROOF. It is straightforward to see that the $(d^2 + 1) \times (d^2 + 1)$ Hessian matrix of Φ can be written in the 2×2 block representation:

$$\nabla \nabla^\top \Phi = P + \lambda Q,$$

where

$$P := \begin{pmatrix} \frac{2}{q^{\frac{2}{d}}} I & -\frac{4}{d} \frac{q' a}{q^{1+\frac{2}{d}}} \\ -\frac{4}{d} \frac{q' a^\top}{q^{1+\frac{2}{d}}} & \frac{2}{d} \left(1 + \frac{2}{d}\right) \frac{|a|^2 q^2}{q^{2+\frac{2}{d}}} \end{pmatrix} \text{ and}$$

$$Q := \begin{pmatrix} 0 & 0 \\ 0 & \frac{2}{q} - 4D \frac{q'}{q^2} + 2(1+D^2) \frac{q'^2}{q^3} \end{pmatrix}.$$

It is trivial to verify that $Q \geq 0$, since Q_{22} is a strictly positive quadratic function of argument D . Since the leading blocks of the matrix P are positive definite and the Schur complement

$$P_{22} - P_{21} P_{11}^{-1} P_{12} = \frac{|a|^2}{q^{2+\frac{2}{d}}} \frac{2}{d} \left(1 - \frac{2}{d}\right) \geq 0$$

is nonnegative definite, overall convexity of Φ is established. \square

REMARK 3. Note that we have just proved the convexity of the function Φ . As an immediate consequence we obtain polyconvexity of the functional (1), because it is a particular case of our functional (4) with $\chi = q$.

Listing 1. A complete L-BFGS-based mesh untangling example

```

1 from mesh import Mesh
2 import numpy as np
3 from scipy.optimize import fmin_l_bfgs_b
4
5 mesh = Mesh() # generate a test quad mesh
6 n = mesh.nverts
7 Q = [ np.matrix('-1,-1;1,0;0,0;1'), np.matrix('-1,0;1,-1;0,1;0,0'), # quadratures for
8       np.matrix('0,0;0,-1;1,1;-1,0'), np.matrix('0,-1;0,0;1,0;-1,1') ] # every quad corner
9
10 def jacobian(U, qc, quad): # evaluate the Jacobian matrix at the given quadrature point
11     return np.matrix([[U[quad[0] ], U[quad[1] ], U[quad[2] ], U[quad[3] ]],
12                      [U[quad[0]+n], U[quad[1]+n], U[quad[2]+n], U[quad[3]+n]]]) * Q[qc]
13
14 for iter in range(10): # outer L-BFGS loop
15     mindet = min( [ np.linalg.det( jacobian(mesh.x, qc, quad) ) for quad in mesh.quads for qc in range(4) ] )
16     eps = np.sqrt(1e-6**2 + .04*min(mindet, 0)**2); # the regularization parameter  $\epsilon$ 
17
18     def energy(U): # compute the energy and its gradient for the map  $\vec{u}$ 
19         F,G = 0, np.zeros(2*n)
20         for quad in mesh.quads: # sum over all quads
21             for qc in range(4): # evaluate the Jacobian matrix for every quad corner
22                 J = jacobian(U, qc, quad)
23                 det = np.linalg.det(J)
24                 chi = det/2 + np.sqrt(eps**2 + det**2)/2 # the penalty function  $\chi(\epsilon, \det(J))$ 
25                 chip = .5 + det/(2*np.sqrt(eps**2 + det**2)) # its derivative  $\chi'(\epsilon, \det(J))$ 
26                 f = np.trace(np.transpose(J)*J)/chi # quad corner shape quality
27                 F += f
28                 dfdj = (2*J - np.matrix([[J[1,1],-J[1,0]],[-J[0,1],J[0,0]]])*f*chip)/chi #  $\frac{\partial f}{\partial a_c}$ : derivative w.r.t the Jacobian
29                 dfdu = Q[qc] * np.transpose(dfdj) # chain rule for the real variables
30                 for i,v in enumerate(quad):
31                     if (mesh.boundary[v]): continue # the boundary verts are locked
32                     G[v ] += dfdu[i,0]
33                     G[v+n] += dfdu[i,1]
34             return F,G
35     mesh.x = fmin_l_bfgs_b(energy, mesh.x)[0] # inner L-BFGS loop
36     print(mesh) # print wavefront .obj file

```

Listing 2. A simplistic quad mesh class

```

1 import numpy as np
2
3 class Mesh():
4     def __init__(self): # generate the test problem: a regular 2d grid with upper half shifted
5         n = 8
6         self.x = [ i/n + int(j>=n//2)*3/5 for j in range(n) for i in range(n) ] + \
7                 [ 2*j/n - int(j>=n//2)*3/5 for j in range(n) for i in range(n) ] # 2D geometry
8         self.quads = [ [i+j*n, i+1+j*n, i+1+(j+1)*n, i+(j+1)*n] for j in range(n-1) for i in range(n-1) ] # connectivity
9         self.boundary = [ i==0 or i==n-1 or j==0 or j==n-1 for j in range(n) for i in range(n) ] # vertex boundary flags
10
11     @property
12     def nverts(self):
13         return len(self.x)//2
14
15     def __str__(self): # wavefront .obj output
16         ret = ""
17         for v in range(self.nverts):
18             ret = ret + ("v %f %f %f\n" % (self.x[v], self.x[v+self.nverts]))
19         for f in self.quads:
20             ret = ret + ("f %d %d %d %d\n" % (f[0]+1, f[1]+1, f[2]+1, f[3]+1))
21         return ret

```