Scalable Locally Injective Mappings

MICHAEL RABINOVICH and ROI PORANNE
ETH Zurich
and
DANIELE PANOZZO
New York University
and
OLGA SORKINE-HORNUNG
ETH Zurich

We present a scalable approach for the optimization of flip-preventing energies in the general context of simplicial mappings, and specifically for mesh parameterization. Our iterative minimization is based on the observation that many distortion energies can be optimized indirectly by minimizing a family of simpler proxy energies. Minimization of these proxies is a natural extension of the local/global minimization of the ARAP energy. Our algorithm is simple to implement and scales to datasets with millions of faces. We demonstrate our approach for the computation of maps that minimize a conformal or isometric distortion energy, both in two and three dimensions. In addition to mesh parameterization, we show that our algorithm can be applied to mesh deformation and mesh quality improvement.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms: Mesh Parameterization, Optimization

Additional Key Words and Phrases: parameterization, bijectivity, scalability

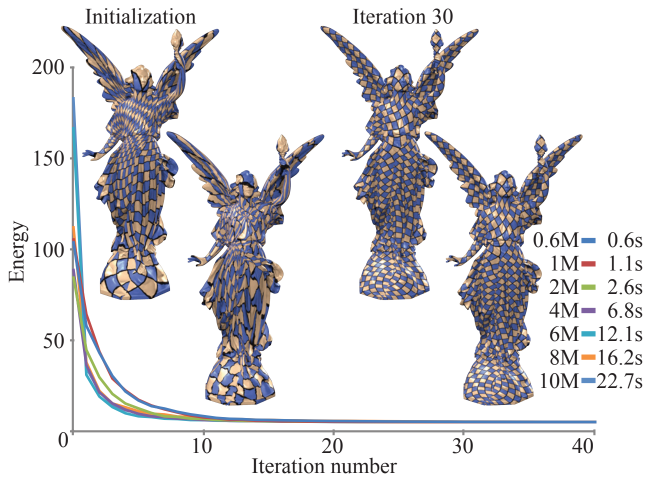


Fig. 1. We compare the behavior of our algorithm on progressively simplified versions of the Lucy model. We observe that the number of iterations required is not dependent on the resolution of the mesh. The time required per iteration in seconds appears on the righthand side of the image.

1. INTRODUCTION

Mappings are an essential tool in computer graphics and geometry processing. One of the most basic uses, and the main focus of this paper, is *mesh parameterization*. Many practical applications, such as texture mapping, remeshing, shape correspondence and attribute transfer rely on the computation of a low-distortion parameterization. The problem has been extensively studied, and a plethora of algorithms have been devised. Linear methods were proposed first, providing efficient ways to compute parameterizations, but only able to ensure injectivity of the map when the mesh boundary is fixed a priori, which induces a high distortion. As more powerful processors became available, nonlinear optimization became tractable, allowing one to compute free boundary, injective or bijective maps of a very high quality. Still, current nonlinear approaches typically require long computation times and do not scale well to large datasets, such as detailed scanned surfaces like the one in Fig. 1.

In this paper, we propose a simple algorithm that combines the benefits of the two approaches: it scales well to large datasets with millions of elements (Fig. 1) and minimizes state-of-the-art nonlinear energies (Fig. 2). In particular, we focus on minimizing *flip-preventing* energies and we propose an algorithm that is guaranteed to produce parameterizations without any flipped elements.

The key idea of our method is to minimize the nonlinear energy using much simpler proxy functions that permit the use of a local/global approach. Our algorithm scales well to large datasets even using a single core, and it can take advantage of the recent developments in parallel solution of linear systems to be deployed on multi-core architectures. While we are unable to provide a strict bound on the convergence rate, we experimentally found that the number of iterations required by our method is related to the geometric surface complexity and is not affected by the tessellation density

This work was supported in part by the ERC grant iModel (StG-2012-306877), a gift from Adobe, and Courant Institute's faculty start-up funding.

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

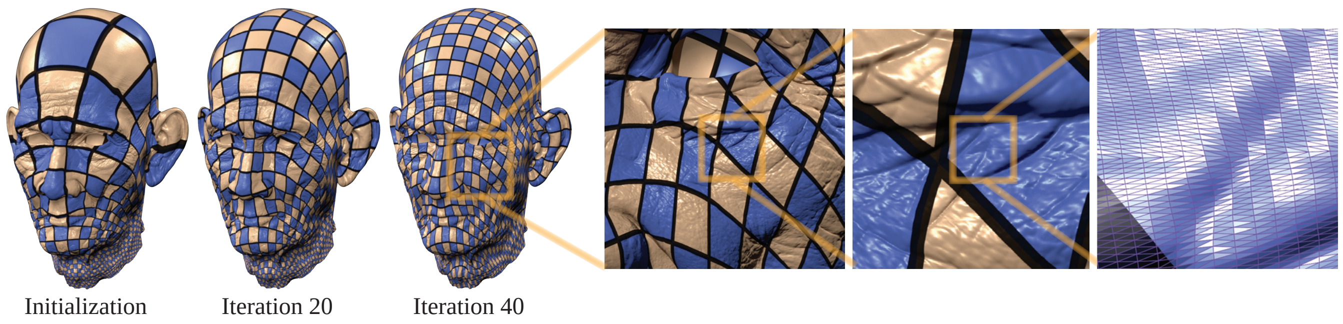


Fig. 2. A locally injective parameterization obtained by minimizing the symmetric Dirichlet energy [Smith and Schaefer 2015] on a mesh with over 25 million triangles, computed with our algorithm in 80 minutes. The algorithm starts from a highly distorted locally injective initialization and in only 40 iterations, each requiring to solve a sparse linear system, it converges to a map with low isometric distortion that is guaranteed to be free of inverted elements.

(Fig. 1). Since each iteration merely requires solving a sparse linear system, massive datasets can be parameterized quickly.

2. PREVIOUS WORK

Mappings are one of the most researched subjects in computer graphics, and specifically the problem of generating locally injective 2D and 3D mappings has garnered a lot of attention in the past decades. In this section, we mention only the most closely related work on the topic of large scale mesh parameterization and we refer to [Floater and Hormann 2005; Sheffer et al. 2006] for in-depth surveys.

Linear methods. Linear methods compute a mesh parameterization by solving a linear system, where each mesh vertex is represented as a weighted average of its neighbors. They have been proposed to parameterize topological disk patches [Tutte 1963] or topological spheres [Aigerman and Lipman 2015]. For topological disks, linear methods can be guaranteed to produce bijective parameterizations if the patch boundary is fixed to a convex shape and the weights are positive [Floater 2003]. Free-boundary methods exist that minimize a measure of conformal distortion [Desbrun et al. 2002; Lévy et al. 2002; Zayer et al. 2007; Ben-Chen et al. 2008; Mullen et al. 2008], but they are not guaranteed to produce a bijective map.

Nonlinear methods. Many nonlinear deformation energies have been proposed in the literature for both conformal and isometric distortion. They are typically minimized using standard optimization methods, such as Newton [Sheffer and de Sturler 2001; Chao et al. 2010; Sanan 2014], Gauss-Newton [Sanan 2014], quasi-Newton [Smith and Schaefer 2015] and second-order cone programming [Aigerman et al. 2014]. To simplify implementation and reduce memory usage, several works [Hormann and Greiner 2000; Labsik et al. 2000; Schreiner et al. 2004] opt for a block descent optimization, where in each iteration only a single vertex is free to move. Similarly, [Levi and Zorin 2014; Fu et al. 2015] optimize an independent subset of vertices in parallel. These methods do not scale to large datasets, since the number of iterations they need grows quickly with the size of the mesh. In contrast, we observe that the number of iterations required by our method is related to the geometric complexity and not the dataset size (Fig. 1): even datasets with millions of elements can be parameterized within a few iterations.

The local/global minimization of isometric distortion [Sorkine and Alexa 2007; Liu et al. 2008] iteratively alternates between a local step and a global step. In the local step, each element is individually perfectly mapped (without any distortion), and in the global step, a linear system is solved to stitch all elements back together. This process very quickly recovers from a bad initialization, but it is slow to converge to a local minimum when it is close to it. The decoupling of a local condition from a global "stitching" has been successfully used in other parameterization algorithms [Weber et al. 2012] and to enforce complex constraints [Bouaziz et al. 2012; Poranne et al. 2013]. Our method uses the local/global paradigm and enriches it with a reweighting scheme to efficiently minimize nonlinear, flip-preventing energies.

Non-flipping invariant. A recent series of works [Schüller et al. 2013; Fu et al. 2015; Smith and Schaefer 2015] have proposed parameterization energies with a term that goes to infinity when an element inverts. These *flip-preventing* energies are minimized starting from a flipless initialization (e.g., [Tutte 1963]) using line search to ensure that they never leave the feasible region. This approach is guaranteed to create a locally injective map given a feasible starting point, but the energies are numerically difficult to optimize, leading to high running times.

Our algorithm is specifically designed to optimize these energies and, differently from all existing methods, quickly recovers from the highly distorted starting point.

Bounding, projections. Another approach to the creation of locally injective maps is based on directly *bounding* the distortion. Similar to the above, a parameterization algorithm is first employed to generate an initial locally injective map. The energy of the map is then optimized while adhering to a specified distortion bound [Lipman 2012; Kovalsky et al. 2014; Sanan 2014; Chen and Weber 2015]. A similar, more recent approach projects any map, possibly with flips, to the closest map that has no flips [Aigerman and Lipman 2013; Kovalsky et al. 2015]. The major limitation with these approaches is that the elements in the solution they generate have suboptimal distortion. In fact, their distortion tends to be the highest possible without violating the bound (Fig. 3). These methods are not guaranteed to find a valid solution [Myles et al. 2014] and they often need thousands of iterations to find one. Instead of projecting into the feasible space, [Fu and Liu 2016] propose to first unfold all simplices separately, and then solve a nonlinear optimization to glue them together. Similarly to the previous methods, they cannot guarantee to produce a valid locally injective map. An interesting special case is the convex energy used in ex-rotated elasticity simulations, for which Liu et al. [2016] propose a numerical approach that can both minimize the energy and robustly recover from flipped elements. However, the ex-rotated energy requires to specify a fixed

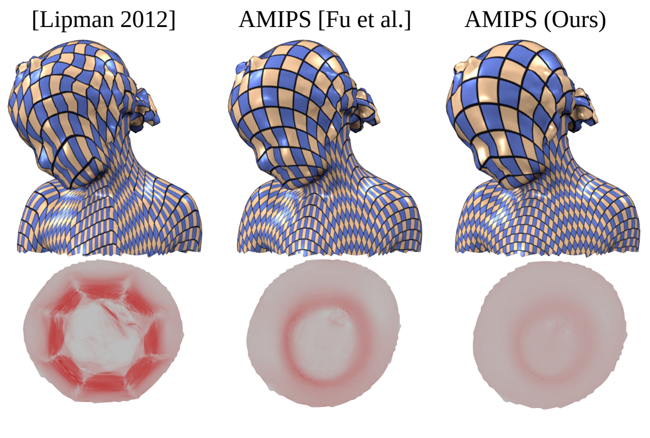


Fig. 3. The bounded ARAP energy (left, result taken from [Fu et al. 2015]) pushes the triangles to the distortion bound, while a direct minimization of the AMIPS energy [Fu et al. 2015] evenly distributes the distortion over the surface (middle). Our approach can also optimize the AMIPS energy (right), further reducing the energy to a local minimum. This model has 11K faces and took our method less than 20 iterations to converge. Bottom row: the distortion magnitude is visualized by the saturation of the red color.

rotation per element as input, inhibiting its use for parameterization tasks and limiting its applicability to a very specific isometric distortion measure [Liu et al. 2016].

Stiffening. Another strategy is the so-called *stiffening*, where the idea is to try and coerce inverted elements to reorient correctly. Examples include [Irving et al. 2004], where an added force acts on inverted elements in a simulation, and [Martin et al. 2011], where a volume term is added to the energy of each element. Another method related to ours is found in [Bommes et al. 2009]: there, the inverted elements are progressively reweighted in the parameterization energy, as opposed to minimizing the original energy in the space of locally injective maps. Finally, the IRLS algorithm [Pighin and Lewis 2007] can be similarly considered as a special form of stiffening. All these methods can be enriched with a line search to ensure the generation of locally injective maps, but they get stuck before reaching a local minimum and thus produce maps that are considerably inferior in quality when compared with the results of our algorithm (Fig. 4).

We discuss these methods in more detail in Sec. 5.

Large scale. Computing locally injective maps with a large number of elements is a challenging numerical problem that has been tackled in surprisingly few research papers in graphics. Apart from linear methods that scale well since they only involve the solution of a linear system, the only other works we are aware of are ABF++ [Sheffer et al. 2005],[Kovalsky et al. 2015], and [Kovalsky et al. 2016]. The first uses a multiresolution hierarchy to make the problem tractable at the expense of a higher distortion compared to the original ABF [Sheffer and de Sturler 2001], and it can only optimize for angle preservation. The algorithm of Kovalsky et al. [2015] is a projection method to the space of conformal maps. It does not minimize map distortion (Fig. 3), it cannot generate isometric parameterizations [Kovalsky 2016] and it often fails to find a valid map, making it impractical for many applications. Concurrently to this work, [Kovalsky et al. 2016] also introduced a scalable method for computing locally injective maps. We compare it with our method in Sec. 5 and demonstrate that our method is significantly faster.

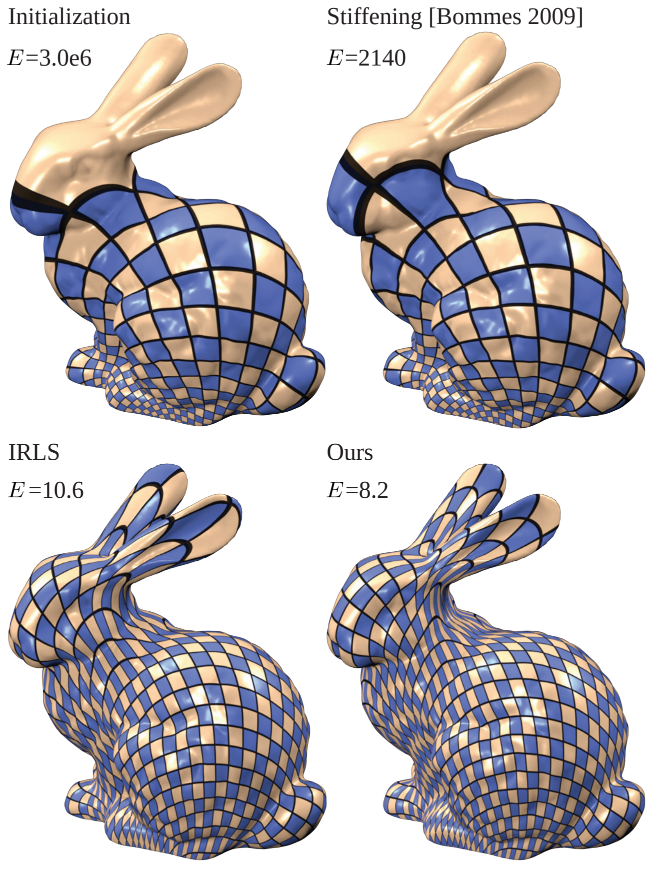


Fig. 4. We compare different numerical methods to minimize the symmetric Dirichlet energy (Eq. (3)) starting from a Tutte's initialization (top left): the greedy stiffening [Bommes et al. 2012] (top right), Iteratively Reweighted Least Squares (bottom left) and our approach (bottom right). See Sec. 5 for a detailed explanation of these techniques.

Other applications. In addition to parameterization, our algorithm can be used to deform 3D objects, and we can also adapt our method to minimize mesh improvement energies, as suggested in [Lipman 2012; Aigerman and Lipman 2013; Fu et al. 2015]. "Seamless" rigid and conformal parameterizations (i.e., parameterizations whose gradients match across seams, see [Myles and Zorin 2012; Myles et al. 2014; Diamanti et al. 2015]) are also supported, although we cannot enforce the integer translations that are necessary for remeshing applications [Bommes et al. 2012].

3. PRELIMINARIES AND PROBLEM STATEMENT

Our main objective is to devise an efficient and scalable algorithm to compute high quality isometric parameterizations. Denote the input triangle mesh by $M = (V, F)$, where V is the set of vertices and F is the set of faces. In our setup, a parameterization $\Phi : M \to \mathbb{R}^{|V|\times 2}$ is a continuous piecewise affine map from M into a planar domain; triangles $f \in F$ are mapped to triangles by an affine map $\Phi|_f$, and the different affine maps agree on the common edges. We denote the mapped coordinates of the vertices by $\mathbf{x} \in \mathbb{R}^{|V|\times 2}$. In the general case, the input mesh cannot be flattened without introducing some geometric distortion. The distortion is quantified by a distortion measure $\mathcal{D}$ that reflects changes between the source and mapped triangles' areas, angles, or a combination of both. The distortion measure $\mathcal{D}$ depends solely on the shape of the source and mapped triangles and is naturally invariant to rotations and translations. Therefore, we assume that each triangle $f \in F$ is equipped with its own arbitrary local isometric mapping to a triangle on the plane, $p_f : \mathbb{R}^3 \to \mathbb{R}^2$. The parameterization mapping of a triangle f can be written as $\Phi|_f = \phi_f \circ p_f$, where $\phi_f : \mathbb{R}^2 \to \mathbb{R}^2$ is an affine map in the plane. Since $\mathcal{D}$ is invariant to translations, it can be formulated solely in terms of the 2×2 Jacobians of each affine transformation ϕ_f, namely $\mathbf{J}_f(\mathbf{x}) := \nabla\phi_f$. Note that $\mathbf{J}_f(\mathbf{x})$ is a linear function of $\mathbf{x}$. Then the energy we wish to minimize is

$$\min_{\mathbf{x}} E(\mathbf{x}) = \sum_{f \in F} A_f\, \mathcal{D}(\mathbf{J}_f(\mathbf{x})), \tag{1}$$

where $\mathcal{D}(\cdot)$ is the distortion measure, and A_f is the area of element f.

We say that $\mathcal{D}$ is an isometric distortion measure if $\mathcal{D}$ is minimal only for rotations. A popular choice for such a distortion measure is the As-Rigid-As-Possible (ARAP) measure [Liu et al. 2008], which is defined by

$$\mathcal{D}_{\mathrm{ARAP}}(\mathbf{J}_f(\mathbf{x})) = \|\mathbf{J}_f(\mathbf{x}) - \mathbf{R}(\mathbf{J}_f(\mathbf{x}))\|_F^2, \tag{2}$$

where $\mathbf{R}(\mathbf{J}_f(\mathbf{x}))$ is the closest rotation to $\mathbf{J}_f(\mathbf{x})$ in the Frobenius norm, and $\|\cdot\|_F$ denotes the Frobenius norm. Minimization of the associated nonlinear energy (1) with the ARAP distortion measure is often achieved with the alternating local/global algorithm [Sorkine and Alexa 2007; Liu et al. 2008]. We review the algorithm in Sec. 4. This local/global optimization has the very interesting property of making large steps in the minimization when initialized with a point far from the solution. This well-known property makes the local/global approach attractive, since a few iterations are sufficient to obtain a result that is good enough for many practical purposes. In addition to parameterization, this technique has been successfully applied to shape deformation [Sorkine and Alexa 2007; Jacobson et al. 2012], architectural design [Bouaziz et al. 2012], and physical simulation [Bouaziz et al. 2014], where its fast progress in the initial iterations is paramount to achieving interactive performance. However, the ARAP energy only softly penalizes degenerate and inverted elements, and they commonly appear in practice [Civit-Flores and Susin 2014; Martinez Esturo et al. 2014], making the resulting parameterizations unusable in many geometry processing tasks, such as texture mapping and remeshing (Fig. 5). This is a direct result of the ARAP energy's bias towards shrinking, in contrast to other distortion measures such as the symmetric Dirichlet [Schreiner et al. 2004; Smith and Schaefer 2015], as seen in Fig. 6. The symmetric Dirichlet energy punishes equally on scaling and shrinking. More precisely, this distortion measure satisfies $\mathcal{D}(\mathbf{J}) = \mathcal{D}(\mathbf{J}^{-1})$ for every $\mathbf{J}$. Our objective is minimizing (1) where $\mathcal{D}$ is the continuous symmetric Dirichlet energy:

$$\mathcal{D}(\mathbf{J}_f(\mathbf{x})) = \begin{cases} \|\mathbf{J}_f(\mathbf{x})\|_F^2 + \|\mathbf{J}_f^{-1}(\mathbf{x})\|_F^2 & \text{if } \det(\mathbf{J}_f(\mathbf{x})) \geq 0 \\ \infty & \text{if } \det(\mathbf{J}_f(\mathbf{x})) < 0 \end{cases} \tag{3}$$

The above definition can be seen as the same energy defined in [Smith and Schaefer 2015], which is infinite for degenerate Jacobians, but continuously extended for negative determinants. Later in the paper we additionally show that we can minimize other distortion measures that are also infinite for degenerate Jacobians, and we similarly extend them to be infinite for orientation flipping Jacobians. Minimizing an energy composed of these distortion measures ensures no triangle degenerates or flips and provides an overall higher quality parameterization, as seen in Figures 5, 6. Our method can be seen as an extension of the local/global algorithm to minimize energies other than ARAP, significantly outperforming current algorithms. Sec. 4 introduces our parameterization algorithm that efficiently minimizes the symmetric Dirichlet energy. Sec. 5 analyzes its performance, comparing it to other optimization methods. In Sec. 6 we further generalize the algorithm to minimize other geometric energies, and in Sec. 7 we demonstrate various applications of our method.

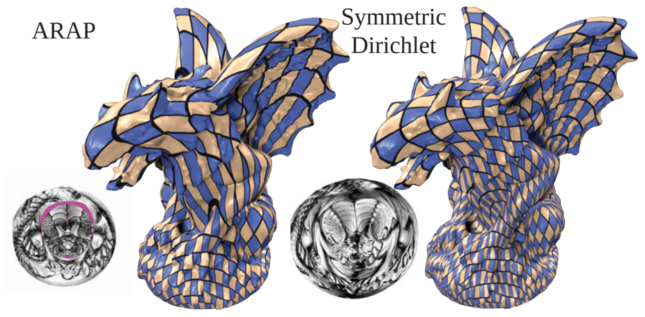


Fig. 5. The ARAP energy (left) introduces 6K inverted triangles in the parameterization (highlighted in magenta), which cause highly distorted regions around the neck and the wing of the Gargoyle. Our algorithm avoids this problem by minimizing a flip-preventing symmetric Dirichlet energy (right). This model has 99K faces and took our algorithm 20 iterations and 2.6 seconds to optimize.

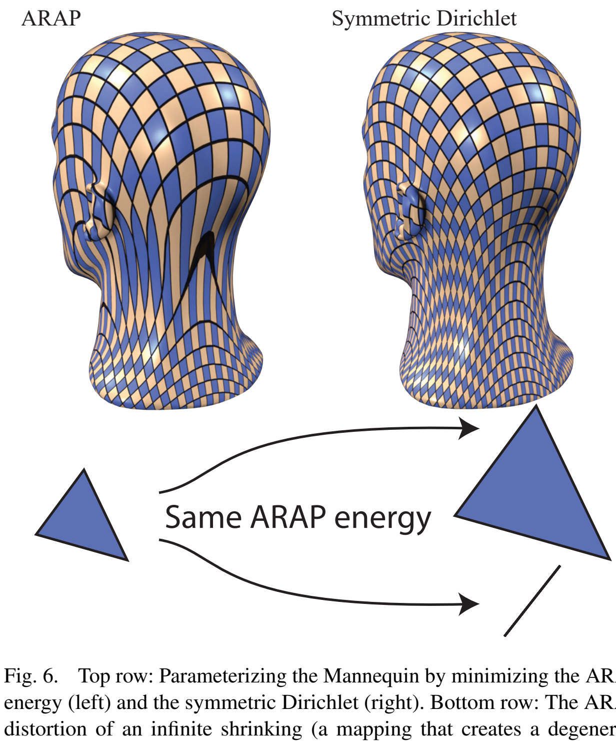


Fig. 6. Top row: Parameterizing the Mannequin by minimizing the ARAP energy (left) and the symmetric Dirichlet (right). Bottom row: The ARAP distortion of an infinite shrinking (a mapping that creates a degenerate triangle) is equal to the distortion of a triangle scaled to less than three times its area. This shrinking bias leads to a parameterization with a lower quality that might contain degenerate or flipped triangles.

4. ISOMETRIC PARAMETERIZATION

We first briefly review the local/global algorithm for minimization of the ARAP distortion measure.

Local/Global optimization. Liu *et al.* [2008] proposed to minimize the ARAP energy using a *local/global* algorithm, an iterative process that alternates between two steps. At iteration k, the local step greedily pushes the Jacobian $\mathbf{J}_f(\mathbf{x})$ of each element $f \in F$ towards the closest minimizer, which is a rotation. That is,

$$\mathbf{J}_f^k := \mathbf{J}_f(\mathbf{x}^{k-1}) \tag{4}$$

$$\mathbf{R}_f^k := \mathbf{R}(\mathbf{J}_f^k) = \mathbf{U}\mathbf{V}^\top \tag{5}$$

where $\mathbf{x}^{k-1}$ is the previous iterate and $\mathbf{J}_f^k = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ is the *signed* SVD of $\mathbf{J}_f^k$. The global step that follows minimizes the deviation from those rotations, essentially averaging the distortion among all elements in order to stitch them together. This is done by solving

$$\mathbf{x}^k = \arg\min_{\mathbf{x}} \sum_{f\in F} A_f \|\mathbf{J}_f(\mathbf{x}) - \mathbf{R}_f^k\|_F^2 \; + \lambda \left\|\mathbf{x} - \mathbf{x}^{k-1}\right\|^2 . \tag{6}$$

The second term is a proximal term that makes the solution unique, removing the invariance of the energy to rigid transformations, and it is equivalent to adding hard constraints to remove the translational and rotational degrees of freedom. We use $\lambda = 10^{-4}$ in this paper. Since the Jacobians are linear functions of $\mathbf{x}$, the above is a simple quadratic minimization that can be computed by solving a linear system (see [Liu et al. 2008]), which is always full rank by construction thanks to the proximal term.

We observe that Eq. (6) minimizes the energy (1) with the ARAP distortion in (2) for a specific set of rotations. We suggest another point of view by introducing a family of proxy functions

$$\mathcal{P}^{\mathbf{R}_f^k}(\mathbf{J}) = \|\mathbf{J} - \mathbf{R}_f^k\|_F^2 \tag{7}$$

and equivalently rewriting Eq. (6) as

$$\mathbf{x}^k = \arg\min_{\mathbf{x}} \sum_{f\in F} A_f \, \mathcal{P}^{\mathbf{R}_f^k}(\mathbf{J}_f(\mathbf{x})) + \lambda \left\|\mathbf{x} - \mathbf{x}^{k-1}\right\| . \tag{8}$$

Note that the proxy in Eq. (7) is almost identical to the ARAP distortion itself in Eq. (2), and they are in fact equal in the current iteration.

To simplify notation, we write $\mathcal{D}(\mathbf{J})$ instead of $\mathcal{D}_{\mathrm{ARAP}}(\mathbf{J})$, and we rewrite the local step as a projection operator from a given Jacobian $\mathbf{J}_f^k$ to the closest minimizer of $\mathcal{D}(\mathbf{J})$:

$$Min(\mathcal{D}) := \{\mathbf{J} \mid \mathbf{J} \text{ is a minimum of } \mathcal{D}\} \tag{9}$$

$$\underset{\mathcal{D}}{\mathrm{Proj}}(\mathbf{J}_f^k) := \arg\min_{\mathbf{R}\in Min(\mathcal{D})} ||\mathbf{J}_f^k - \mathbf{R}||_F^2 = \mathbf{R}_f^k \tag{10}$$

We further observe that this convex quadratic proxy has the following properties:

$$\textit{Majorizer:} \quad \mathcal{P}^{\mathbf{R}_f^k}(\mathbf{J}) \geq \mathcal{D}(\mathbf{J}) \quad \forall \mathbf{J} \tag{11}$$

$$\textit{Matching gradients:} \quad \nabla_{\mathbf{J}}\mathcal{P}^{\mathbf{R}_f^k}(\mathbf{J}_f^k) = \nabla_{\mathbf{J}}\mathcal{D}(\mathbf{J}_f^k) \tag{12}$$

$$\textit{Closest minimizer:} \quad \arg\min_{\mathbf{J}} \mathcal{P}^{\mathbf{R}_f^k}(\mathbf{J}) = \underset{\mathcal{D}}{\mathrm{Proj}}(\mathbf{J}_f^k) \tag{13}$$

Property (11) guarantees that the proxy energy is at least as high as the original ARAP energy, and since it is exactly equal at the given point, minimizing the proxy is guaranteed to reduce the ARAP energy at every step. Property (12) ensures that the energy (6) *locally* approximates ARAP. Indeed, $\mathbf{J}_f(\mathbf{x})$ is a linear function of $\mathbf{x}$, and the following also holds:

$$\nabla_{\mathbf{x}}\mathcal{P}^{\mathbf{R}_f^k}(\mathbf{J}_f^k) = \nabla_{\mathbf{x}}\mathcal{D}(\mathbf{J}_f^k). \tag{14}$$

Property (13) is more of a global nature. The nonlinear ARAP distortion measure is minimal for all rotations, while the quadratic proxy has only a single minimizer. This property states that the proxy at each iteration is minimal only for the rotation that is closest to $\mathbf{J}_f^k$. Generalizing the local/global algorithm for the symmetric Dirichlet distortion measure means finding an appropriate class of proxy distortion measures with the properties (11)-(13).

Sadly, this is not possible. As any quadratic function is finite on a finite interval, finding a *quadratic* proxy distortion measure with the majorizer property (11) is impossible for the symmetric Dirichlet, or any other distortion measure that has an infinite value for degenerate or orientation flipping Jacobians. Therefore, at best, we can hope to have a proxy distortion measure with the matching gradient (12) and closest minimizer (13) properties.

Local-Global as a descent algorithm. Let us rewrite the local/global algorithm as a descent algorithm. The local step remains the same (Eq. (5)), and then the global step generates a search direction $\mathbf{d}^k$:

$$\mathbf{d}^k = \tilde{\mathbf{x}}^k - \mathbf{x}^{k-1}, \tag{15}$$

where $\tilde{\mathbf{x}}^k$ is computed as in Eq. (6) and $\mathbf{x}^{k-1}$ is the previous iterate. Then the algorithm simply takes a full step in the search direction:

$$\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha\,\mathbf{d}^k, \;\; \alpha = 1, \tag{16}$$

and the iterations are continued for the next value $k := k+1$. Since the proxy is convex quadratic and the gradients match, $\mathbf{d}^k$ must point towards a descent direction for Eq. (2). This allows us to use a variation of the local/global algorithm with a step size $\alpha \neq 1$ and a line search. Consequently, property (12) is sufficient to minimize an energy of the form (1) composed of a general distortion measure $\mathcal{D}$.

Weighted proxy functions. The challenge in extending the local/global descent algorithm to a more general energy $\mathcal{D}$ is to find a class of proxy functions $\mathcal{P}$ that satisfy similar conditions to Eq. (12), (13). We first note that for the symmetric Dirichlet distortion, as well as any other isometric distortion, the set of minimizers is the same, namely the set of rotations. The Euclidean projection of a Jacobian $\mathbf{J}_f^k$ to the closest minimum is in fact the closest rotation, exactly as in ARAP. This leads us to propose the following family of proxy functions:

$$\mathcal{P}_{\mathbf{W}}^{\mathbf{R}}(\mathbf{J}) = \|\mathbf{W}(\mathbf{J} - \mathbf{R})\|_F^2 , \tag{17}$$

where $\mathbf{W}$ is a 2×2 matrix. These proxy functions satisfy (13) by construction, and we wish to find a matrix $\mathbf{W}$ such that (12) holds. The proxy functions are similar to the ARAP proxy (Eq. (7)), with the only difference being the term $\mathbf{W}$, which is the affine transformation required to warp the proxy to locally match the gradient of the distortion energy. This enhanced proxy energy is quadratic (with fixed $\mathbf{R}$ and $\mathbf{W}$) and can be rewritten in matrix form and minimized (see details in Appendix A).

Given $\mathbf{x}^{k-1}$ and $\mathbf{R}_f^k$, we can find a weight matrix $\mathbf{W}_f^k$, such that Eq. (12) holds for $\mathcal{P}_{\mathbf{W}_f^k}^{\mathbf{R}_f^k}$:

$$\nabla_{\mathbf{J}_f} \left\|\mathbf{W}_f^k(\mathbf{J}_f - \mathbf{R}_f^k)\right\|_F^2 = \nabla_{\mathbf{J}_f}\mathcal{D}(\mathbf{J}_f), \text{ at } \mathbf{J}_f = \mathbf{J}_f(\mathbf{x}^{k-1}). \tag{18}$$

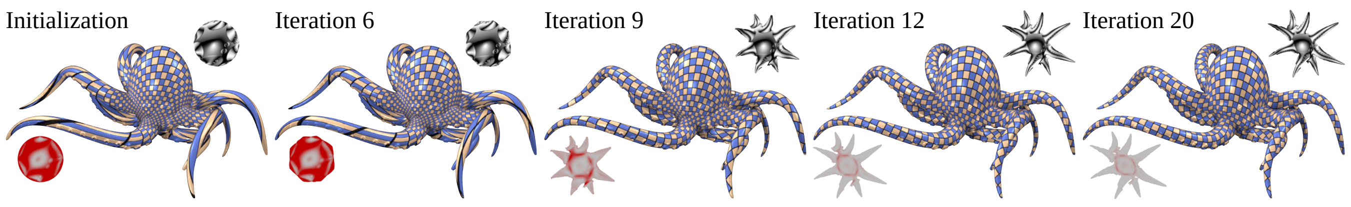


Fig. 7. Minimization of the symmetric Dirichlet energy on the Octopus model. Note how quickly the boundary of the UV map recovers from the distorted starting point. This model has 299K faces and took 20 iteration and 5.5 seconds to optimize.

For brevity, we drop the indices and simply write Eq. (18) as $\nabla_{\mathbf{J}} \|\mathbf{W}(\mathbf{J}-\mathbf{R})\|_F^2 = \nabla_{\mathbf{J}} \mathcal{D}(\mathbf{J})$. We can then expand the l.h.s.:

$$\nabla_{\mathbf{J}} \|\mathbf{W}(\mathbf{J}-\mathbf{R})\|_F^2 = \\ \nabla_{\mathbf{J}} \mathrm{tr}(\mathbf{W}^\top \mathbf{W}(\mathbf{J}-\mathbf{R})(\mathbf{J}-\mathbf{R})^\top) = \\ (\mathbf{W}^\top \mathbf{W} + \mathbf{W}\mathbf{W}^\top)(\mathbf{J}-\mathbf{R}).$$

Assuming $\mathbf{J}-\mathbf{R}$ is invertible, we can manipulate this equation and find that $\mathbf{W}$ must satisfy

$$\mathbf{W}^\top \mathbf{W} + \mathbf{W}\mathbf{W}^\top = \nabla_{\mathbf{J}} \mathcal{D}(\mathbf{J})\,(\mathbf{J}-\mathbf{R})^{-1}. \tag{19}$$

If $(\mathbf{J}-\mathbf{R})^{-1}$ does not exist, we take the pseudo-inverse instead. Since the left-hand side is positive semidefinite, a solution to Eq. (19) exists if and only if the right hand-side is positive semidefinite, which is always the case for isometric energies such as symmetric Dirichlet (see Sec. 6 for more details). The unique solution can be found explicitly:

$$\mathbf{W} = \left(\frac{1}{2}\nabla_{\mathbf{J}} \mathcal{D}(\mathbf{J})\,(\mathbf{J}-\mathbf{R})^{-1}\right)^{1/2}, \tag{20}$$

where the square root is a matrix principal root, which we compute via diagonalization. Alternatively, this equation can be rewritten to sidestep the diagonalization to improve performance, as detailed in Sec. 6 (see Eq. (28)).

Note that $\nabla_{\mathbf{J}} \mathcal{D}_{\mathrm{ARAP}}(\mathbf{J}) = 2(\mathbf{J}-\mathbf{R})$, reducing the previous equation to $\mathbf{W} = \mathbf{I}$. Thus, our algorithm reduces to [Liu et al. 2008] when used on $\mathcal{D}_{\mathrm{ARAP}}$.

Line search. The proxy energies generate good descent directions during the minimization: at each step our algorithm alternates the computation of optimal rotations (local step) with the minimization of the proxy function to find a descent direction. Similarly to Eq. (8), we use the weighted proxy and solve,

$$\mathbf{p}^k := \arg\min_{\mathbf{x}} \sum_{f \in F} A_f\, \mathcal{P}_{\mathbf{W}_f^k}^{\mathbf{R}_f^k}(\mathbf{J}_f(\mathbf{x})) + \lambda \left\|\mathbf{x}-\mathbf{x}^{k-1}\right\|. \tag{21}$$

Then we use $\mathbf{p}^k$ to define the search direction,

$$\mathbf{d}^k := \mathbf{p}^k - \mathbf{x}^{k-1} \tag{22}$$

and we use this direction for the next step,

$$\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha\,\mathbf{d}^k,$$

where α is the step size determined by a line search that minimizes the *original* energy (1), *without* the proximal term. Starting from a locally injective map (Tutte's embedding) and using a line search to minimize any energy that is infinite exactly when elements degenerate or flip ensures that the generated map is flip-free. In practice, we use a line search strategy similar to [Smith and Schaefer 2015] to choose an α such that we never cross the point where an element flips. At such points the energy is infinite. We thus initially set $\alpha = \min\{0.8\alpha_{\max}, 1\}$, where $\alpha_{\max}$ is the maximal step size with no foldovers. We then proceed with a bisection line search, where the initial interval is $[0, \alpha_{\max}]$, until we find a point that satisfies Wolfe's conditions (see [Nocedal and Wright 2006], section 3.4). Our algorithm is guaranteed to generate a descent direction at every iteration, and therefore decrease the energy at each iteration. We have found experimentally that our algorithm always converges to a local minimum. However we lack a formal convergence proof, and this remains an open question. This is also the case for the original local/global ARAP algorithm. We mention that by the Zoutendijk theorem ([Nocedal and Wright 2006] section 3.2), such a proof could be found by bounding the angle between $\mathbf{d}^k$ and the gradient of the energy, which is directly related to the condition number of our linear system.

Reweighted local/global for symmetric Dirichlet. Enriching the local/global algorithm with the matrix reweighting scheme and line search leads to our algorithm, which is simple to implement and scales to datasets with millions of elements. Algorithm 1 sketches the pseudocode of our method. We provide the source code of a reference implementation in *libigl* [Jacobson et al. 2016], as well as an optimized parallel optimization on GitHub [Rabinovich 2016]. We analyze the method's properties in Sec. 5 and we show how it can be easily extended to various distortion energies and to 3D locally injective maps in Sec. 6.

Algorithm 1: Reweighted local/global

```
Input:
    A mesh $M$ with a set of vertices $V$ and elements $F$
Output:
    A set of mapping coordinates $\mathbf{x}$ minimizing Eq. (1)

Initialization:
$\mathbf{x}^0$ = Tutte($V$, $F$)
Optimization:
for $k$ = 1 to max_iterations do
    Compute closest rotation $\mathbf{R}_f^k$ for each Jacobian $\mathbf{J}_f(\mathbf{x}^{k-1})$
    Update the weights $\mathbf{W}_f^k$ for each $f \in F$ using Eq. (20)
    Solve Eq. (21) to find $\mathbf{p}^k$
    Set $\mathbf{d}^k$ using Eq. (22)
    Find $\alpha_{\max}$, as in [Smith and Schaefer 2015](Section 3.3)
    Perform bisection line search to find step size $\alpha$, starting
    from $\alpha = \min\{1, 0.8\alpha_{\max}\}$ in the interval $[0, \alpha_{\max}]$
    $\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha\,\mathbf{d}^k$
```

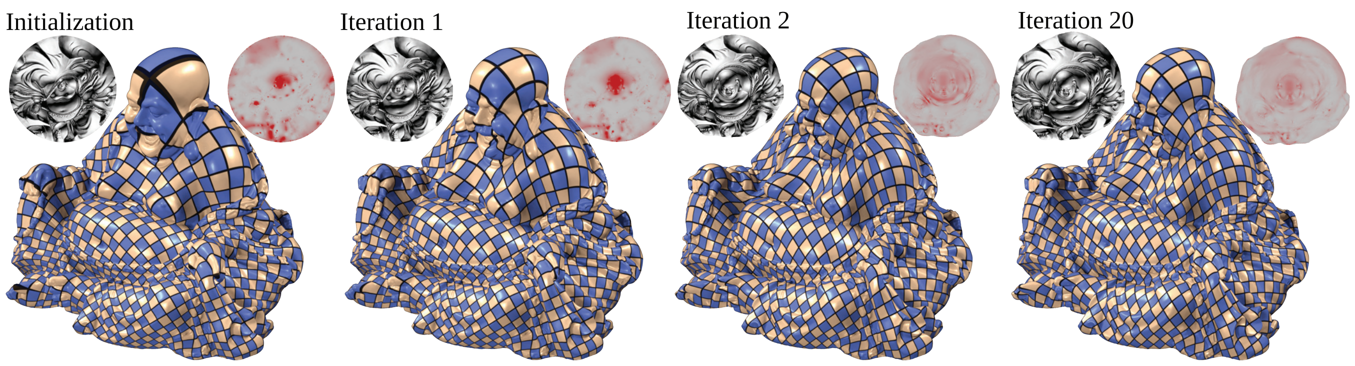


Fig. 8. Minimization of the symmetric Dirichlet energy on the Buddha model. Despite the massive the dataset (470K faces), our algorithm produces an optimized locally injective parameterization in 14 seconds.

5. ANALYSIS

We apply our algorithm to a series of challenging parameterization problems to observe its properties and scalability to large datasets. We also provide a comparison against Newton descent and L-BFGS [Nocedal and Wright 2006] to demonstrate that our algorithm outperforms classical optimization techniques, and against more recent state of the art methods.

Qualitative convergence behavior. We observed that the convergence behavior in our case is similar to local/global optimization of the ARAP energy, that is, the algorithm progresses very rapidly at the beginning, but then slows down when close to a minimum (see Figures 7, 8, 9). Uneven meshing does not alter the performance of our method (Fig. 10). We demonstrate this by parameterizing two copies of the same surface, one with an homogeneous triangulation and another with an uneven triangulation with hundreds of slivers.

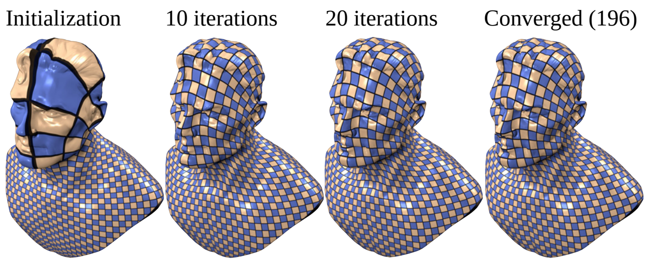


Fig. 9. Our algorithm quickly progresses towards the minimum of the energy. If sufficiently many iterations are executed, our algorithm numerically converges to a minimum, but the differences in quality compared to earlier iterations are negligible. This figure shows the result after 10, 20, and finally, 196 iterations, where the optimization has converged to a minimum.

Our algorithm is extremely robust, and it succeeds even when initialized with a map made of slivers, as demonstrated in Fig. 19 — the resulting parameterization after 20 iterations is very close to being isometric, as can be seen in the texture, even if the algorithm needs many more iterations to reach a numerical minimum. Newton's method is much slower, and only after 2000 iterations does the quality of the parameterization start to resemble ours.

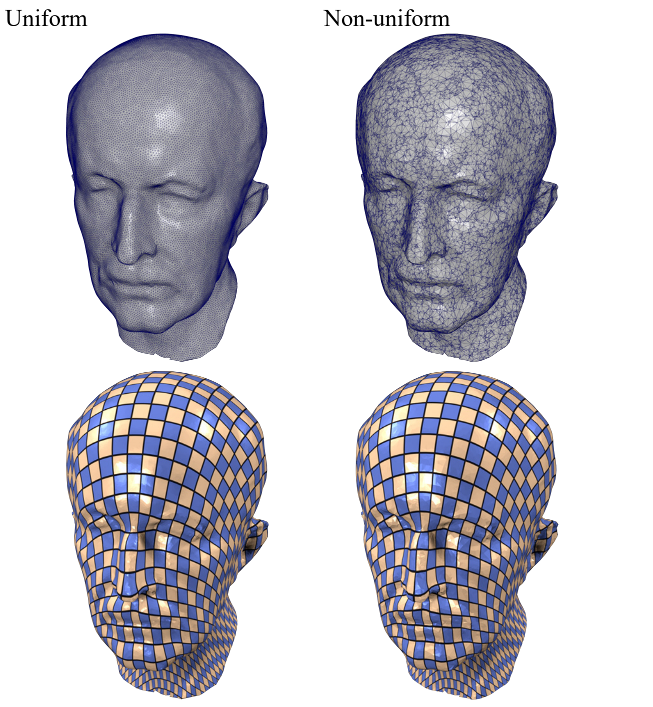


Fig. 10. Our algorithm is robust and consistently produces high quality locally-injective maps in as little as 20 iterations, even in presence of low-quality triangulations. In this example we show the results of running our parameterization process on the same shape with uniform and non-uniform triangulations. As can be seen, our algorithm reaches similar minima regardless of the mesh quality. In this case, the non-uniform triangulation has smallest triangle angle of 0.222 degrees and the largest triangle angle of 178.05. The max triangle area divided by the smallest is 94,618.

Comparison with standard numerical methods. In Fig. 12 (top) we compare our iterations against L-BFGS. Since the L-BFGS iterations are faster than ours, we plot the time on the x-axis to ensure a fair comparison. Our method is considerably faster and produces a high quality map in a fraction of the time. In Fig. 12 (bottom), we compare against Newton's method. This method can get stuck if the Hessian is not positive semidefinite (PSD). A common remedy in simulation applications is to regularize the Hessian by

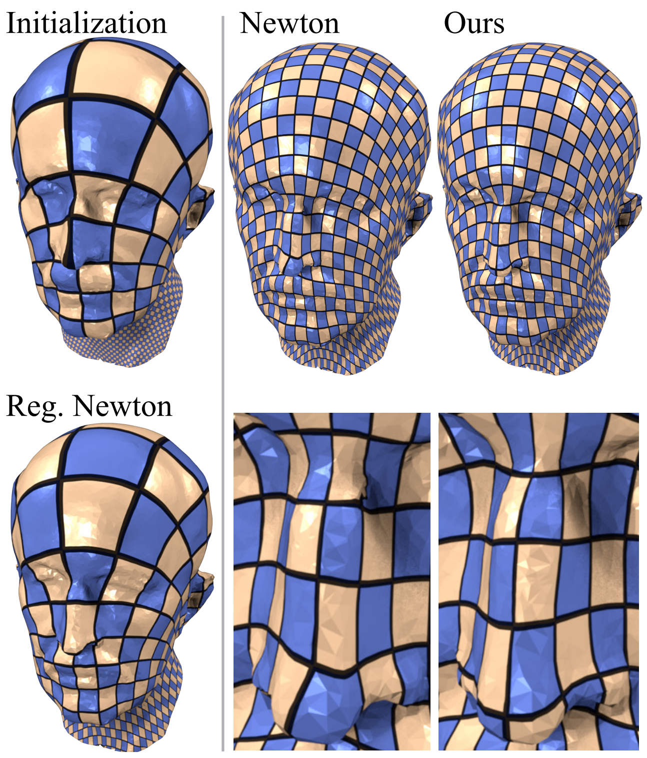


Fig. 11. Our algorithm outperforms Newton's method when applied to non-uniform triangulations. In this case, Newton's method stopped before reaching a minimum, while the regularized Newton did very little progress after 500 iterations.

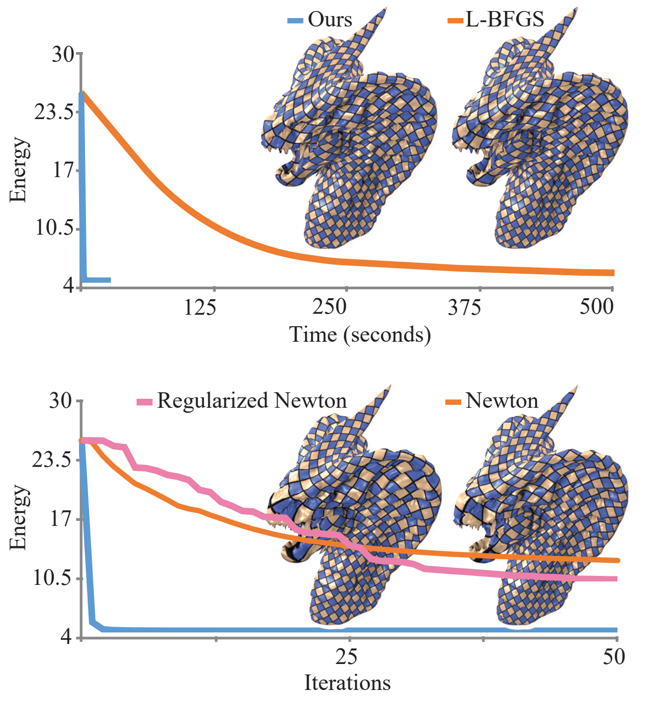


Fig. 12. Our iterations are slower than those of L-BFGS, since we have to solve a sparse linear system, but overall, they progress much faster (top). Our iterations are also making considerably more progress than Newton iterations when we are far from the minimum (bottom). This model has 386K faces, and in both cases we minimize the symmetric Dirichlet energy. See Fig. 19 for another comparison with Newton's method.

projecting the Hessian of each face on the set of PSD matrices (see e.g. [Teran et al. 2005]). We observe that for parameterization it is not clear which method performs better, as cases vary. We report the results of both the normal and the regularized version of Newton's method. In both cases, our algorithm is considerably faster. Another example of this behavior is shown in Fig. 11 and in Fig. 19.

Comparison with similar methods. The Accelerated Quadratic Proxy (AQP) algorithm of [Kovalsky et al. 2016] minimizes a distortion energy by iteratively computing descent directions using a modified Newton's method, where instead of the true Hessian, which changes at each iteration, they use the cotangent Laplacian, which remains fixed. These directions are later fed into a line search algorithm with a customized acceleration scheme. The descent directions are generated by solving

$$\mathbf{d}^k = -\mathbf{A}^{-1}\mathbf{g}(\mathbf{x}_{k-1}), \tag{23}$$

where $\mathbf{g}(\mathbf{x})$ is the gradient of the objective, and $\mathbf{A}$ incorporates the Laplacian and additional positional constraints. A similar method by [Martin et al. 2013] uses different weighted sums of powers of the Laplacian, which represent highpass, bandpass and lowpass filters. We show in Appendix C that our method can be written in a similar way. For both approaches above, the matrix $\mathbf{A}$ is fixed and can be prefactored, leading to fast iterations. This is not the case for our method, where the matrix $\mathbf{A}$ changes in every iteration, so that only symbolic factorization can be precomputed. Nevertheless, our algorithm substantially outperforms these methods since it generates superior descent directions, resulting in fewer iterations and a lower running time. We compare our method against our implementation of the approach of [Kovalsky et al. 2016] in Fig. 13 and in Table I. The code of the two algorithms uses the PARDISO solver, and the same optimized line search and energy evaluation code. Our method is between 2 to 6 times faster on simple datasets, and up to 14 times faster on challenging models such as the Bunny and the Gargoyle.

The method of [Martin et al. 2013] does not scale to large models. We implemented the algorithm of [Martin et al. 2013] for mesh parameterization with the Symmetric Dirichlet energy, using the same optimized code base and solvers. Our method is more than 200 times faster on the 11K faces Bimba mesh, and more than 2,100 times faster on the 99K faces Lion Vase model.

Scalability. The efficiency of our method stems from the experimental observation that the number of iterations is related to the geometric complexity of the model, instead of depending on the density of the tessellation. We show evidence supporting this claim in Fig. 1, where we plot the energy on a progressive input mesh (Lucy), sampled with 0.6-6 million faces. Note that the majority of the competing methods become impractically slow for models larger than 100K vertices. There are only two exceptions: (1) [Kovalsky et al. 2015], which does not minimize a distortion energy, often fails to find a solution and cannot be used to create isometric parameterizations [Kovalsky 2016]; and (2) [Kovalsky et al. 2016], which is considerably slower than our approach, as demonstrated in our experiments (Table I).

Why not ARAP with Line search? Our approach relies on the line search proposed by [Smith and Schaefer 2015] to ensure the local injectivity of the map. It is tempting to simply add the line search to e.g. ARAP parameterization algorithm,

Table I. Comparison with [Kovalsky et al. 2016]

Model (Figure)	#Faces	#Vertices	Our iterations	AQP iterations	Our time (s)	AQP time (s)	Speedup factor
Bimba (3)	11K	5.6K	10 / 20	131 / 145	0.3 / 0.55	1.48 / 1.63	4.93 / 2.96
Bunny (4)	108K	54K	10 / 20	923 / 979	1.98 / 3.47	44.06 / 46.73	22.25 / 13.46
Gargoyle (5)	99K	49K	10 / 20	901 / 946	1.49 / 2.53	34.92 / 36.6	23.43 / 14.46
Octopus (7)	300K	151K	10 / 20	222 / 298	3.42 / 5.48	22.82 / 29.68	6.67 / 5.41
Superman (9)	190K	95K	10 / 20	217 / 284	2.9 / 4.83	17.98 / 23.09	6.2 / 4.78
Buddha (8)	470K	235K	10 / 20	87 / 117	7.98 / 13.5	24.36 / 31.15	3.05 / 2.3
Max Planck (10)	84K	42K	10 / 20	81 / 102	1.11 / 1.83	3.1 / 3.73	2.79 / 2.03
Dragon Head (12)	387K	194K	10 / 20	316 / 404	6.57 / 11.33	61.31 / 77	9.33 / 6.79
Bear (14)	296K	148K	10 / 20	181 / 212	5.22 / 9	24.83 / 28.72	4.75 / 3.19
Lion Vase (15)	99K	49K	10 / 20	239 / 268	1.48 / 2.5	8.79 / 9.76	5.93 / 3.9

Comparison of mesh parameterization minimizing the Symmetric Dirichlet energy with our implementation of [Kovalsky et al. 2016], called Accelerated Quadratic Proxy (AQP). We ran our algorithm for a fixed number of iterations (10 and 20), and compared both the number of iterations and the time taken for the AQP method to reach the same energy. The linear system of the AQP method can be numerically prefactored, while in our algorithm we can only perform symbolic factorization. Though each iteration of our algorithm is more expensive, our proxy generates better descent directions and our algorithm is between 2 to 14 times faster. The prefactoring of the linear systems is included in the timings.

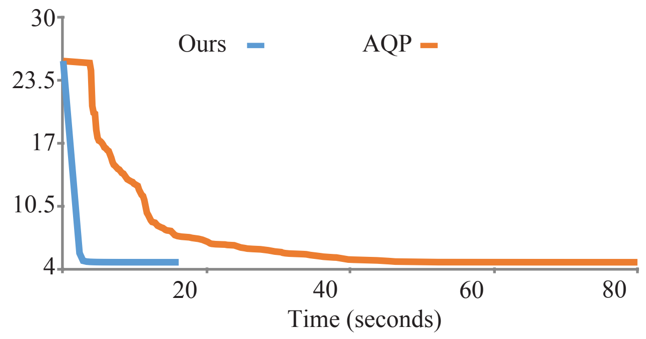
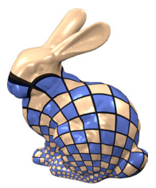


Fig. 13. Comparison of our algorithm and AQP for computing a parameterization minimizing the Symmetric Dirichet energy of the 386K faces Dragon Head model. Our approach is significantly faster, especially at the first iterations.

obtaining a method that is also guaranteed to produce locally injective maps. However, the quality of the map will be poor because the line-search will lock the algorithm whenever one triangle degenerates, and further progress will not be allowed.



In the inset, we repeat the experiment shown in Fig. 4 using the ARAP parameterization algorithm with the the line search proposed by [Smith and Schaefer 2015]. The algorithm is not able to make much progress, producing a result with a much higher distortion than ours (Fig. 4).

Relationship to IRLS and greedy stiffening. Our approach is loosely related to the stiffening methods, which are often used in global parameterization [Bommes et al. 2009], where Eq. (6) is modified to include a per-element scalar weight w_f, i.e.,

$$\min_{\mathbf{x}} \sum_{f \in F} A_f \, w_f^k \|\mathbf{J}_f(\mathbf{x}) - \mathbf{R}_f^k\|_F^2. \tag{24}$$

The rationale is to increase this weight for the elements that flip, hoping that this will lead to a locally injective map. This approach is a heuristic that is not guaranteed to succeed, since increasing the weight of an element to remove a flip could cause a flip in a neighboring element, and this could (and does in practice) repeat indefinitely. We tried using this approach in our formulation, replacing our matrix weights with a scalar weight and updating it using the update rule proposed in [Bommes et al. 2009], but this is not guaranteed to generate a descent direction, and the algorithm gets stuck, as shown in Fig. 4.

Another related algorithm is iterative reweighted least squares (IRLS), which also uses scalar weights for the purpose of locally matching the energy value. This is sufficient to ensure convergence for the specific case of L^p norms. We tried to adapt this idea to our framework, using scalar weights and the update rule proposed in [Yoshizawa et al. 2004; Pighin and Lewis 2007]:

$$w_f^k = \frac{\mathcal{D}(\mathbf{J}_f(\mathbf{x}^{k-1}))}{\|\mathbf{J}_f(\mathbf{x}^{k-1}) - \mathbf{R}_f^k\|_F^2}. \tag{25}$$

If, in a given iteration, an element is mapped to an almost degenerate one, the symmetric Dirichlet energy in the numerator becomes huge, while the proxy in the denominator does not, and so in the next iteration the IRLS scalar weight strategy penalizes more on this element by giving it a larger weight. This could prevent it from flipping in the next iteration, and one could expect it to produce a result that is closer to the actual energy that we would like to minimize. If an element is scaled, we could expect the weight to be lower than 1, since the proxy tends to penalize it too much. However, our experiments show that this update rule (which turned out to be conceptually similar to the stiffening of [Bommes et al. 2009]) is not sufficient to guarantee convergence, and in fact eventually often fails to find a descent direction (Fig. 4).

Intuitively, the expressivity of scalar weights is limited, for instance when a triangle is stretched very heavily along just one axis, while not being stretched along the other axis. In this case, a single scalar weight per-triangle cannot distinguish between the axes, and both of them are penalized the same. A more proper choice is to penalize each axis separately, and this can only be expressed by using matrix weights. We elaborate on this in the next section.

6. GENERAL DISTORTION ENERGIES

We proceed with the treatment of specific isometric distortion measures, as well as generalizing to other types of geometric distortion measures. From now on, we assume that $\mathcal{D}(\mathbf{J})$ is any rotation invariant distortion measure, as we define below. Note that any $\mathcal{D}(\mathbf{J})$ that solely measures geometric changes is in fact rotation invariant. Additionally, we fill the gap that remains from the previous section, that is, ensuring that Eq. (19) is indeed valid, and that its right hand side is positive semidefinite. We start with the latter, as it naturally leads to the derivation of the relevant formulas.

SVD viewpoint. A *geometric* distortion measure is *rotation invariant*:

Definition 1. $\mathcal{D}(\mathbf{J})$ is *rotation invariant* if

$$\mathcal{D}(\mathbf{J}) = \mathcal{D}(\mathbf{U}\mathbf{J}\mathbf{V}^\top)$$

for any rotation matrices $\mathbf{U}$ and $\mathbf{V}$.

All rotation invariant distortion measures can be written solely in terms of the singular values of $\mathbf{J}$, as shown in the following lemma.

LEMMA 2. *Let* $\mathbf{J} = \mathbf{U}\mathbf{S_J}\mathbf{V}^\top$ *be the singular value decomposition of* $\mathbf{J}$ *(we use* $\mathbf{S_A}$ *to denote the diagonal matrix containing the singular values of* $\mathbf{A}$*). Then,*

$$\mathcal{D}(\mathbf{J}) = \mathcal{D}(\mathbf{S_J}), \tag{26}$$

$$\nabla_{\mathbf{J}}\mathcal{D}(\mathbf{J}) = \mathbf{U}\,\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})\,\mathbf{V}^\top. \tag{27}$$

Proof. See Appedix B.1.

For example, the ARAP distortion measure, which is rotation invariant, can be written as

$$\|\mathbf{J}-\mathbf{R}\|_F^2 = \|\mathbf{S_J}-\mathbf{I}\|_F^2 = \sum_{i=1}^{d}(\sigma_i - 1)^2,$$

where σ_i are the singular values of $\mathbf{J}$.

Using Lemma 2, we can write Eq. (20) in SVD form:

$$\mathbf{W} = \mathbf{U}\left(\frac{1}{2}\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})(\mathbf{S_J}-\mathbf{I})^{-1}\right)^{1/2}\mathbf{U}^\top = \mathbf{U}\mathbf{S_W}\mathbf{U}^\top. \tag{28}$$

This expression shows that the matrix weights in fact act as scalar weights per singular value of each Jacobian, that is, a single weight-per-axis. Since the matrix expression inside the square root is diagonal, it is trivial to take the root. From Eq. (28) we can clearly see that

$$\nabla_{\mathbf{S}}\mathcal{D}(\mathbf{S_J})(\mathbf{S_J}-\mathbf{I}) \succeq 0 \tag{29}$$

is required in order to take the root. In other words, $\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})(\mathbf{S_J}-\mathbf{I})$ must be PSD, which is always the case for isometric distortion measures, as we show in the following.

Isometric energies. We applied our algorithm to a variety of isometric distortion measures. In our context, a *true* isometric distortion measure is one that is rotationally invariant, minimal *only* for rotations, and is separable in terms of the singular values. In other words, a true isometric distortion measure can be written as

$$\mathcal{D}(\mathbf{S_J}) = \mathcal{D}(\sigma_1,\ldots,\sigma_d) = \sum_i f_i(\sigma_i), \tag{30}$$

where $f_i(\sigma_i)$ convex and minimal when $\sigma_i = 1$. This condition makes it is easy to show that Eq. (29) holds.

Without loss of generality, we demonstrate the solution to Eq. (28) using the symmetric Dirichlet energy. The same steps can be used for any other isometric distortion measures (see Table II and Figure 14). The symmetric Dirichlet energy in terms of the singular values is

$$\mathcal{D}(\mathbf{J}) = \|\mathbf{J}\|_F^2 + \|\mathbf{J}^{-1}\|_F^2 = \sum_{i=1}^{d}(\sigma_i^2 + \sigma_i^{-2}).$$

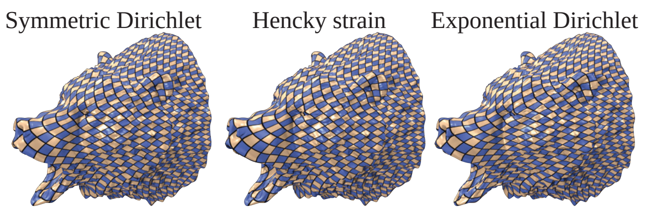


Fig. 14. Minimizing isometric distortions for the Bear model. Our approach is general and supports many distortion energies such as Symmetric Dirichlet (left), Hencky strain (middle) and Exponential Dirichlet (right). This model has 296K faces and required an average of 9 seconds to optimize.

Hence,

$$(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_i = 2(\sigma_i - \sigma_i^{-3}), \tag{31}$$

where we introduce the notation $(\mathbf{D})_i$ to refer to the i'th diagonal entry of the diagonal matrix $\mathbf{D}$. Thus, plugging Eq. (31) in Eq. (28) we obtain

$$(\mathbf{S_W})_i = \sqrt{\frac{\sigma_i - \sigma_i^{-3}}{\sigma_i - 1}} \tag{32}$$

when $\sigma_i \neq 1$, and $(\mathbf{S_W})_i = 4$, which is the limit, otherwise. The expression under the root is always nonnegative, so we can always find weights to match the gradients.

General distortion measures. To extend this construction to energies that are rotation invariant, but not true isometric distortions (i.e., do not satisfy Eq. (30)), we need to slightly change our algorithm. Consider for example the AMIPS energy $E^\star_{iso}$ from [Fu et al. 2015] (see the paper for the reasoning behind this definition), that is defined using the distortion measure

$$\mathcal{D}^\star_{iso}(\mathbf{J}) = \exp\left(s\cdot\mathcal{D}_{iso}(\mathbf{J})\right) \tag{33}$$

where

$$\mathcal{D}_{iso}(\mathbf{J}) = \frac{1}{2}\left[\left(\frac{\mathrm{tr}(\mathbf{J}^\top\mathbf{J})}{\det(\mathbf{J})}\right) + \frac{1}{2}(\det(\mathbf{J}) + \det(\mathbf{J}^{-1}))\right]. \tag{34}$$

Algorithm 1 fails to minimize this energy, because (29) might not hold, and it thus may be impossible to compute weights that locally match the gradients. To address this issue, we generalize the local step (Eq. (5)) by replacing the closest rotation $\mathbf{R}$ with another matrix Λ, which depends on the energy that we wish to minimize. Note that the local step is only used in the proxy energy, which becomes

$$\mathcal{P}^{\Lambda}_{\mathbf{W}}(\mathbf{J}) = \|\mathbf{W}(\mathbf{J}-\Lambda)\|_F^2\,. \tag{35}$$

The modification of the local step changes the step computation Eq. (21) and the formula for $\mathbf{W}$ in Eq. (28), replacing the closest rotation and the identity with the matrices Λ and $\mathbf{S}_\Lambda$. The global step now solves

$$\mathbf{p}^k := \arg\min_{\mathbf{x}} \sum_{f\in F} A_f\,\mathcal{P}^{\Lambda_f^k}_{\mathbf{W}_f^k}(\mathbf{J}_f(\mathbf{x})) + \lambda\left\|\mathbf{x}-\mathbf{x}^{k-1}\right\|, \tag{36}$$

the weight matrix can be computed as

$$\mathbf{W} = \mathbf{U}\left(\frac{1}{2}\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})(\mathbf{S_J}-\mathbf{S}_\Lambda)^{-1}\right)^{1/2}\mathbf{U}^\top. \tag{37}$$

,

and the requirement in Eq. (29) becomes

$$\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})(\mathbf{S_J}-\mathbf{S}_\Lambda) \succeq 0. \tag{38}$$

The condition on the local step is now clear: The sign of each element of $\mathbf{S_J} - \mathbf{S}_\Lambda$ must match the sign of each element of $\nabla_{\mathbf{S_J}} \mathcal{D}(\mathbf{S_J})$. The computation in the local step can then be easily adapted on a case-by-case basis. Note that this construction is a generalization of [Liu et al. 2008].

General construction. Given an arbitrary distortion energy, it is easy to find a matrix Λ that satisfies Eq. (38). For the special (and very general) case of *separably strictly convex* energies, we found a simple construction.

Definition 3. A function $\mathcal{D}(x_1, ..., x_n)$ is *separably strictly convex*, if for each i the single variable function $\mathcal{D}_{(x_1,..,x_{i-1},x_{i+1},..,x_n)}(x_i)$, which is constructed by freezing all of the other variables, is *strictly convex*.

In this case, we can set each $(\mathbf{S}_\Lambda)_i$ to be the solution of

$$\frac{\partial}{\partial \sigma_i} \mathcal{D}_{(\sigma_1,\ldots,\sigma_{i-1},\sigma_{i+1},\ldots,\sigma_d)}(\sigma_i) = 0. \qquad (39)$$

In other words, we set $\mathbf{S}_\Lambda$ such that each entry of $\mathbf{S}_\Lambda$ minimizes the corresponding entry of $\mathcal{D}(\mathbf{S_J})$, assuming the other singular values are fixed. We show in Appendix B.2 that this choice always satisfies Eq. (38). The reasoning behind this choice is to always push each singular value towards its closest minimum. It is also a direct generalization of the isometric case shown above, as the choice of the closest rotation exactly satisfies Eq. (39).

As an example, we derive the expressions for the AMIPS energy (Eq. (33)) in Table II and show the resulting parameterization in Fig. 3. The complete algorithm is summarized in Algorithm 2.

Conformal distortions. While the above construction is general and guaranteed to create weights that match the gradients, it might be suboptimal for certain energies. For example, if we consider the conformal distortion energy proposed (for arbitrary dimension d) in [Fu et al. 2015]

$$\mathcal{D}(\mathbf{J}) = \frac{\operatorname{tr}(\mathbf{J}^\top \mathbf{J})}{\det(\mathbf{J})^{2/d}}, \qquad (40)$$

our general construction produces a matrix in the local step that is not a similarity. We experimentally observed an increase in performance by defining $\Lambda = \bar{\sigma} U V^\top$, where $\bar{\sigma}$ is a scalar, i.e., by finding the closest conformal transformation. We show in Appendix B.3 how to compute $\bar{\sigma}$ for the 2D and 3D case. Similarly to [Fu et al. 2015], we can also minimize the exponential of this energy (Fig. 17).

7. APPLICATIONS

We ran our experiments on a 12-core Xeon clocked at 2.7 GHz, using the PARDISO solver [Schenk et al. 2007; Schenk et al. 2008; Kuzmin et al. 2013] for the linear system solve. We report the running times in Table III, using both a single- and multi-core implementation. The sparsity pattern of the linear system in every iteration never changes, allowing us to reuse the symbolic factorization between iterations. Our method requires a feasible, i.e., inversion free starting point: for 2D, we use Tutte's parameterization with cotangent weights; if they produce a flipped element, we resort to uniform weights which are guaranteed to give us a valid starting point. For the mesh improvement and deformation examples in 3D, the rest pose is used as the starting point.

Single-patch 2D parameterization. Single-patch 2D parameterization is ubiquitously used in modeling software to generate UV maps given a predefined set of cuts. Our approach improves upon existing algorithms, providing superior quality and higher efficiency, in addition to supporting extremely detailed models with millions of elements. We show the quality of our results on several meshes with different energies throughout the paper. We highlight that our method is not tied to a specific parameterization energy, and can minimize all rotation invariant energies, provided that the local step satisfies Eq. (38).

Algorithm 2: Generalized Reweighted local/global

```
Input:
    A mesh $M$ with a set of vertices $V$ and elements $F$
Output:
    A set of mapping coordinates $\mathbf{x}$ minimizing Eq. (1)

Initialization:
$\mathbf{x}^0$ = Tutte($V$, $F$)
Optimization:
for $k$ = 1 to max_iterations do
    Compute the SVD $\mathbf{U S_J V}^\top = \mathbf{J}_f(\mathbf{x}^{k-1})$ for each face $f$
    Update $\Lambda_f^k = \mathbf{U S}_\Lambda \mathbf{V}^\top$, taking $\mathbf{S}_\Lambda$ from Table II
    Update the weights $\mathbf{W}_f^k$ for each $f \in F$ using Eq. (37)
    Solve Eq. (36) to find $\mathbf{p}^k$
    Set $\mathbf{d}^k$ using (22)
    Find $\alpha_{\max}$, as in [Smith and Schaefer 2015](Section 3.3)
    Perform bisection line search to find step size $\alpha$, starting
    from $\alpha = \min\{1, 0.8\alpha_{\max}\}$ in the interval $[0, \alpha_{\max}]$
    $\mathbf{x}^k = \mathbf{x}^{k-1} + \alpha\,\mathbf{d}^k$
```

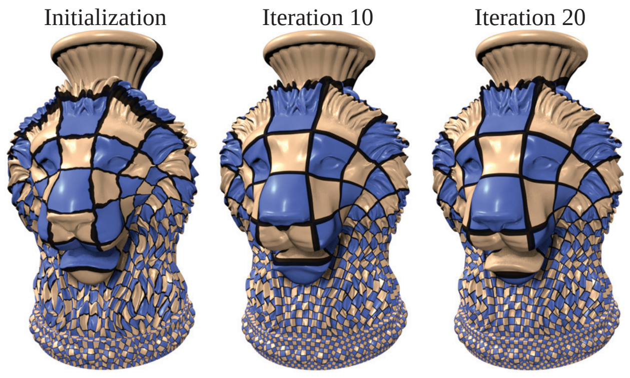


Fig. 15. Minimization of a conformal energy using our method. Note that visually the difference between 10 and 20 iterations is already quite small, and after 20 it becomes negligible.

Seamless 2D parameterization. "Seamless" parameterizations, i.e., parameterizations whose derivatives agree on the seams up to a rotation of multiples of $\pi/2$, are commonly used for remeshing purposes [Bommes et al. 2012]. Our algorithm can generate them by adding compatibility constraints [Bommes et al. 2009] to the seams, as shown in Fig. 16. Note that we currently do not support integer optimization in our framework and we thus cannot produce integer-grid maps [Bommes et al. 2013].

3D deformation. Volumetric deformation energies can be minimized with our method, benefiting in a similar way as 2D parameterization. In Fig. 17, we demonstrate an example of a cube deformation with two different discretization resolutions (48K and

Table II. Energies we used in this paper, expressed also in terms of the singular values, with their derivatives and our choice for the local step. The entries of this table are used in Algorithm 2.

Name	$\mathcal{D}(\mathbf{J})$	$\mathcal{D}(\sigma)$	$(\nabla_{\mathbf{S}}\mathcal{D}(\mathbf{S}))_i$	$(\mathbf{S}_\Lambda)_i$
Symmetric Dirichlet	$\|\mathbf{J}\|_F^2 + \|\mathbf{J}^{-1}\|_F^2$	$\sum_{i=1}^n(\sigma_i^2 + \sigma_i^{-2})$	$2(\sigma_i - \sigma_i^{-3})$	1
Exponential Symmetric Dirichlet	$\exp(s(\|\mathbf{J}\|_F^2 + \|\mathbf{J}^{-1}\|_F^2))$	$\exp(s\sum_{i=1}^n(\sigma_i^2 + \sigma_i^{-2}))$	$2s(\sigma_i - \sigma_i^{-3})\exp(s(\sigma_i^2 + \sigma_i^{-2}))$	1
Hencky strain	$\left\|\log \mathbf{J}^\top\mathbf{J}\right\|_F^2$	$\sum_{i=1}^n(\log^2\sigma_i)$	$2(\frac{\log\sigma_i}{\sigma_i})$	1
AMIPS	$\exp(s\cdot\frac{1}{2}(\frac{\mathrm{tr}(\mathbf{J}^\top\mathbf{J})}{\det(\mathbf{J})} + \frac{1}{2}(\det(\mathbf{J}) + \det(\mathbf{J}^{-1})))$	$\exp(s(\frac{1}{2}(\frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}) + \frac{1}{4}(\sigma_1\sigma_2 + \frac{1}{\sigma_1\sigma_2}))$	$s\cdot\exp(s\cdot(\frac{1}{4}(\sigma_{i+1} - \frac{1}{\sigma_{i+1}\sigma_i^2}) + \frac{1}{2}(\frac{1}{\sigma_{i+1}} - \frac{\sigma_{i+1}}{\sigma_i^2}))$	$\sqrt{\frac{2\sigma_{i+1}^2+1}{\sigma_{i+1}^2+2}}$
Conformal AMIPS 2D	$\frac{\mathrm{tr}(\mathbf{J}^\top\mathbf{J})}{\det(\mathbf{J})}$	$\frac{\sigma_1^2+\sigma_2^2}{\sigma_1\sigma_2}$	$\frac{1}{\sigma_{i+1}} - \frac{\sigma_{i+1}}{\sigma_i^2}$	$\sqrt{\sigma_1\sigma_2}$
Conformal AMIPS 3D	$\frac{\mathrm{tr}(\mathbf{J}^\top\mathbf{J})}{\det(\mathbf{J})^{\frac{2}{3}}}$	$\frac{\sigma_1^2+\sigma_2^2+\sigma_3^2}{(\sigma_1\sigma_2\sigma_3)^{\frac{2}{3}}}$	$\frac{-2\sigma_{i+1}\sigma_{i+2}(\sigma_{i+1}^2+\sigma_{i+2}^2-2\sigma_i^2)}{(3\sigma_i\sigma_{i+1}\sigma_{i+2})^{\frac{5}{3}}}$	$\sqrt{\frac{\sigma_1^2+\sigma_3^2}{2}}$

Table III. Size of the input datasets and running times with and without multi-core parallelization.

Model (Figure)	**#Faces**	**#Vertices**	**Time (s) Multi core**	**Time (s) Single core**
Bimba (3)	11K	5.6K	< 1	< 1
Bunny (4)	108K	54K	3.47	9.8
Gargoyle (5)	99K	49K	2.53	6.5
Octopus (7)	300K	151K	5.48	18.4
Superman (9)	190K	95K	4.83	14.6
Buddha (8)	470K	235K	13.5	43.2
Max Planck (10)	84K	42K	1.83	6.2
Dragon head (12)	387K	194K	11.33	36.8
Bear (14)	296K	148K	9	25.2
Vase Lion (15)	99K	49K	2.5	6.2

250K tetrahedra) and two different distortion energies (exponential Dirichlet and exponential of the conformal AMIP). The running time of our algorithm is 0.5 and 8 seconds per iteration, respectively, and we used 10 iterations.

Mesh improvement. Maps can be used to improve the quality of meshes, and we compare our algorithm against two recent methods [Aigerman and Lipman 2013; Fu et al. 2015] in Fig. 18. We use the exponential Dirichlet energy, which is inspired by the exponential AMIPS energy proposed in [Fu et al. 2015]. As can be seen in Table IV, our approach outperforms the competing methods in all cases except one. We performed 10 iterations for all models, which on average took 3 seconds each.

8. LIMITATIONS AND FUTURE WORK

We presented a general approach to simply and efficiently minimize practical distortion energies commonly used in geometry processing. The major theoretical limitation and advantage of our approach are both inherited from the local/global method: our algorithm is extremely fast while approaching a local minimum but requires many iterations to converge to a numerical minimum. This problem stems from the slow propagation of the rotations in the local step, making a small rotation over a large part of the parameterization hard to recover from. This is however not a practical limitation, since the difference between the result we obtain after 20 iterations

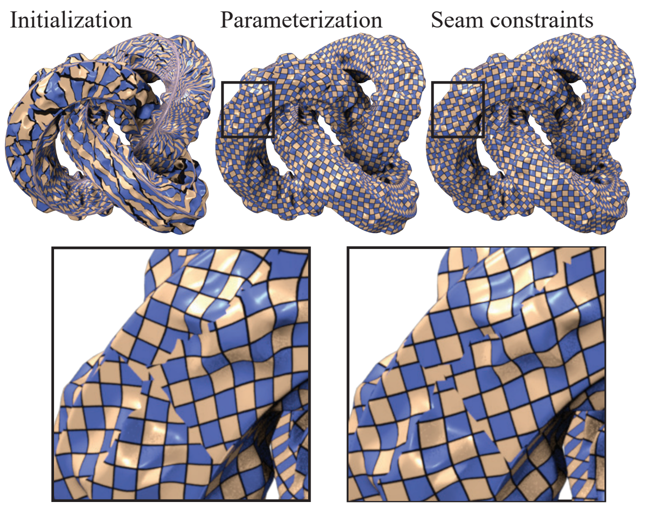


Fig. 16. An example of "seamless" global parameterization computed with our method. Starting from Tutte's embedding (left), we minimize the symmetric Dirichlet energy first, and then activate seamless soft constraints to make the parameterization's derivatives match on the seams, up to permutation.

and the converged result is negligible (Figures 9, 15). On the other hand, the fast progress of our method in the first few iterations is an extremely valuable property for many applications (Fig. 12).

We exemplify this property in a stress test similar to [Smith and Schaefer 2015] (see Fig. 19). The challenge in this test is to recover from Tutte's embedding of a Hilbert-curve-shaped developable surface, which is flipless, but highly distorted. We observed that our first iteration makes the same progress as around 2000 Newton iterations — by combining the two algorithms, we reconstruct the Hilbert curve in less than 200 iterations.

Our algorithm requires a locally injective initialization, which we construct using Tutte's embedding in 2D, but we are not aware of any general construction for 3D maps. This limitation is ameliorated by the fact that maps are mostly used for deformations in 3D, where the rest shape itself is a perfectly valid initial map which we directly use as a locally injective initialization.

A final limitation is that our proxy energy definition only works for rotation invariant distortion energies. It is an interesting venue

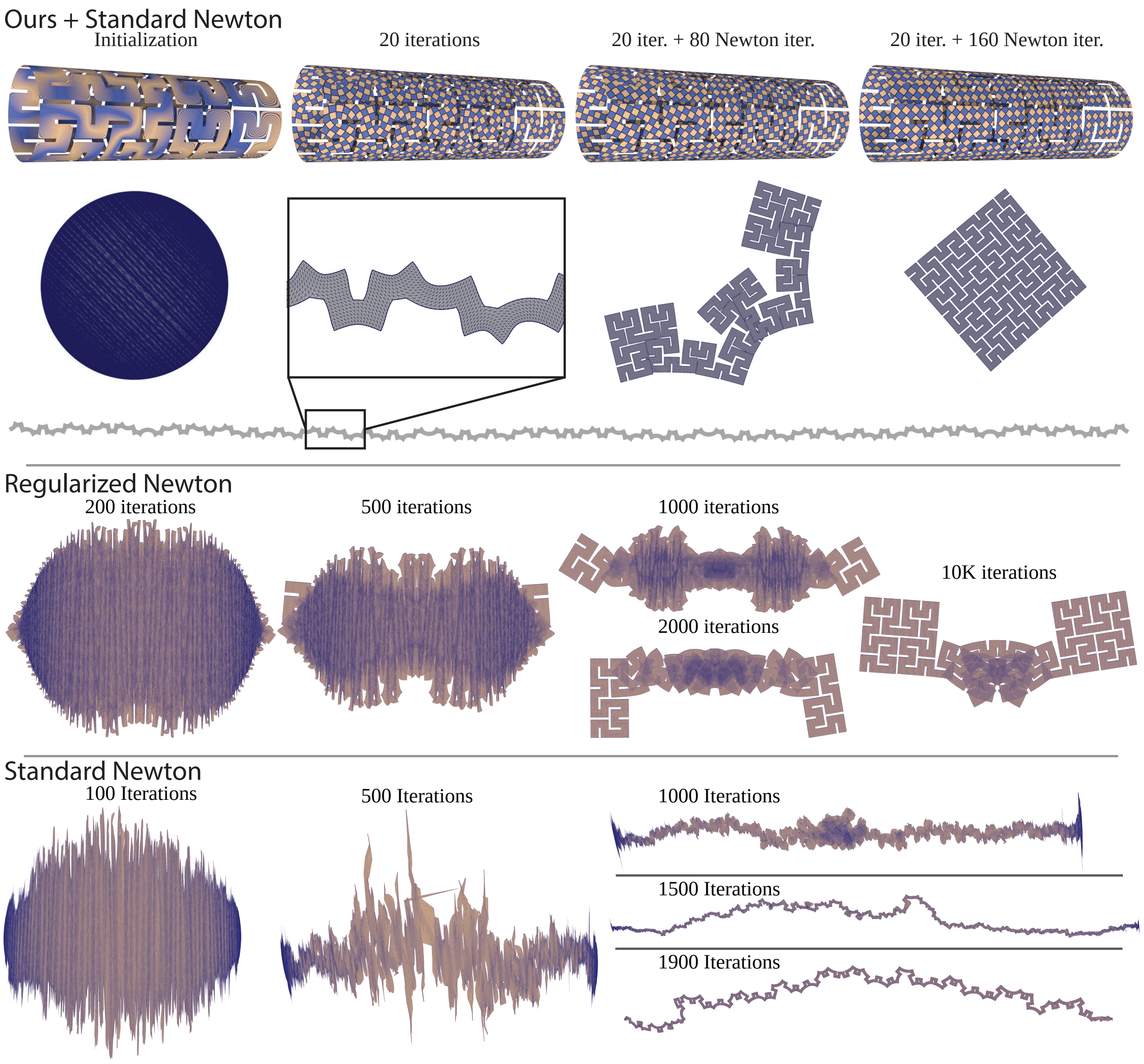


Fig. 19. Example of running our algorithm on a stress test. The highly distorted Tutte's embedding is quickly unrolled by our algorithm in about 20 iterations, producing a map close to being isometric. However, from this point on the progress of our algorithm slows down considerably, although the minimum is still reached after ca. 500K iterations. The regularized Newton method is much slower (middle row) and the system is severely ill-conditioned due to the extremely distorted triangles. The standard, non-reguralized Newton fails to find a descent direction after 1900 iterations. By combining our method with the standard Newton's method, we are able to reach the global minimum in a total of 180 iterations. Note that we used Newton iterations with our method as an initialization *only* in this figure. All the other results in the paper are produced using solely our algorithm.

for future work to extend our local/global method to a wider family of energies, such as those used in finite element simulations.

Acknowledgments

The authors would like to thank Shahar Kovalsky for providing the source code for AQP and for illuminating discussions, and the anonymous reviewers for their useful comments and suggestions. This work was supported in part by the ERC grant iModel (StG-2012-306877), a gift from Adobe, and Courant Institute's faculty start-up funding.

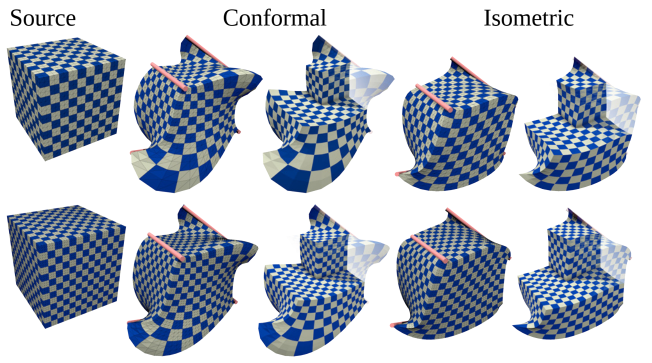


Fig. 17. Our method can be used to deform tetrahedral meshes, minimizing a conformal energy (middle) or the exponential symmetric Dirichlet isometric energy (right). The cubes have 48K (top) and 350K (bottom) tetrahedra, and our algorithm took 5 and 80 seconds, respectively. We picked 4 edges of the cube (shown as cylinders) and manipulated them. The right image in each pair shows the interior of the deformed cube.

Table IV. Comparison of mesh improvement.

Name	Init. Dihed.	BD	AMIPS	Our Method
Duck	(10,163)	(16,148)	(19.6,161.5)	(19, 138.16)
Elephant	(8,167)	(16,148)	(13.7,161.3)	(20.2,141.2)
Elephant2	(15,157)	(18,147)	(19,150)	(23.1,142.3)
Hand	(9,162)	(16,148)	(18,156)	(21.1, 143.3)
Max	(21,151)	(14,153)	(27.2,137.3)	(29,141.5)
Rocker	(10,163)	(16,148)	(21.6,148.7)	(22.8,139.4)
Skull	(0.8,178)	(14,153)	(21.1,147.6)	(17.4,157.8)
Dragon	(31,140)	(28,139)	(27.8,139.37)	(31.8, 137.6)

Comparison of mesh improvement achieved by running [Aigerman and Lipman 2013] (BD) and [Fu et al. 2015] (AMIPS). In each entry in the table we show the minimal and maximal dihedral angle, where the second column shows the initial values. As can be seen, in all cases except for the Skull dataset, our method outperforms the competing methods.

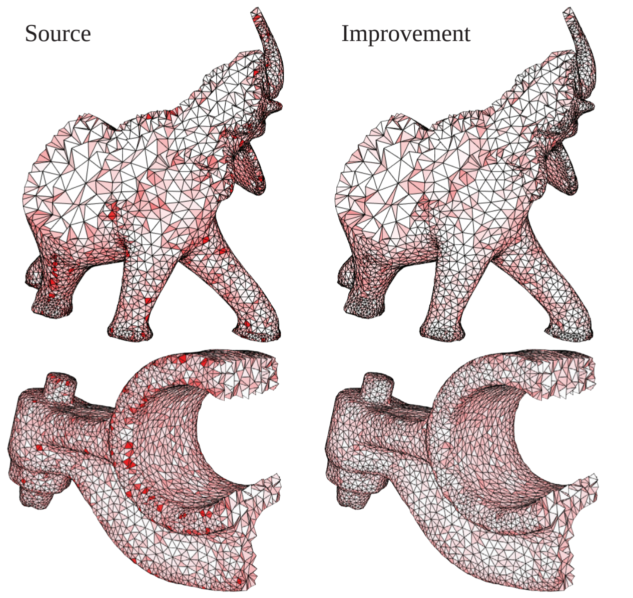


Fig. 18. An example of 3D mesh improvement computed with our method minimizing the Exponential Dirichlet energy. The elephant has 33.5K tetrahedra and the rocker arm 35.5K. Our entire optimization took 3 and 3.2 seconds, respectively.

REFERENCES

AIGERMAN, N. AND LIPMAN, Y. 2013. Injective and bounded distortion mappings in 3D. *ACM Trans. Graph. 32,* 4.

AIGERMAN, N. AND LIPMAN, Y. 2015. Orbifold Tutte embeddings. *ACM Trans. Graph. 34,* 6.

AIGERMAN, N., PORANNE, R., AND LIPMAN, Y. 2014. Lifted bijections for low distortion surface mappings. *ACM Trans. Graph. 33,* 4, 69:1–69:12.

BEN-CHEN, M., GOTSMAN, C., AND BUNIN, G. 2008. Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum 27,* 2, 449–458.

BOMMES, D., CAMPEN, M., EBKE, H.-C., ALLIEZ, P., AND KOBBELT, L. 2013. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32,* 4 (July), 98:1–98:12.

BOMMES, D., LÉVY, B., PIETRONI, N., PUPPO, E., SILVA, C., TARINI, M., AND ZORIN, D. 2012. State of the art in quad meshing. In *Eurographics STARs*.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28,* 3, 77:1–77:10.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-Up: Shaping discrete geometry with projections. *Comput. Graph. Forum 31,* 5, 1657–1667.

BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33,* 4 (July), 154:1–154:11.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph. 29,* 4, 38:1–38:6.

CHEN, R. AND WEBER, O. 2015. Bounded distortion harmonic mappings in the plane. *ACM Trans. Graph. 34,* 4.

CIVIT-FLORES, O. AND SUSIN, A. 2014. Robust treatment of degenerate elements in interactive corotational FEM simulations. *Computer Graphics Forum 33,* 6, 298–309.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. *Comput. Graph. Forum 21,* 3.

DIAMANTI, O., VAXMAN, A., PANOZZO, D., AND SORKINE-HORNUNG, O. 2015. Integrable PolyVector fields. *ACM Trans. Graph. 34,* 4, 38:1–38:12.

FLOATER, M. S. 2003. One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation 72,* 242, 685–696.

FLOATER, M. S. AND HORMANN, K. 2005. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*. Mathematics and Visualization. 157–186.

FU, X. AND LIU, Y. 2016. Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics (SIGGRAPH Asia) 35,* 6.

FU, X.-M., LIU, Y., AND GUO, B. 2015. Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph. 34,* 4.

GILES, M. 2008. An extended collection of matrix derivative results for forward and reverse mode automatic differentiation. Tech. rep., Oxford University Computing Laboratory. Jan.

HORMANN, K. AND GREINER, G. 2000. MIPS: An efficient global parametrization method. In *Curve and Surface Design*. Innovations in Applied Mathematics. 153–162.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. Eurographics Symposium on Computer Animation*. 131–140.

JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations. *ACM Trans. Graph. 31,* 4, 77:1–77:10.

JACOBSON, A., PANOZZO, D., ET AL. 2016. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

KOVALSKY, S. 2016. Private communication.

KOVALSKY, S. Z., AIGERMAN, N., BASRI, R., AND LIPMAN, Y. 2014. Controlling singular values with semidefinite programming. *ACM Trans. Graph. 33,* 4.

KOVALSKY, S. Z., AIGERMAN, N., BASRI, R., AND LIPMAN, Y. 2015. Large-scale bounded distortion mappings. *ACM Trans. Graph. 34,* 6, 191:1–191:10.

KOVALSKY, S. Z., GALUN, M., AND LIPMAN, Y. 2016. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph. 35,* 4, 134.

KUZMIN, A., LUISIER, M., AND SCHENK, O. 2013. Fast methods for computing selected elements of the Green's function in massively parallel nanoelectronic device simulations. In *Proc. Euro-Par*. Lecture Notes in Computer Science. 533–544.

LABSIK, U., HORMANN, K., AND GREINER, G. 2000. Using most isometric parametrizations for remeshing polygonal surfaces. In *Proc. Geometric Modeling and Processing*. 220–228.

LEVI, Z. AND ZORIN, D. 2014. Strict minimizers for geometric optimization. *ACM Trans. Graph. 33,* 6, 185:1–185:14.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21,* 3, 362–371.

LIPMAN, Y. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31,* 4, 108:1–108:13.

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. J. 2008. A local/global approach to mesh parameterization. In *Proc. Symposium on Geometry Processing*. 1495–1504.

LIU, T., GAO, M., ZHU, L., SIFAKIS, E., AND KAVAN, L. 2016. Fast and robust inversion-free shape manipulation. *Comput. Graph. Forum 35,* 2.

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-based elastic materials. *ACM Trans. Graph. 30,* 4, 72:1–72:8.

MARTIN, T., JOSHI, P., BERGOU, M., AND CARR, N. 2013. Efficient non-linear optimization via multi-scale gradient filtering. *Computer Graphics Forum 32,* 6, 89–100.

MARTINEZ ESTURO, J., RÖSSL, C., AND THEISEL, H. 2014. Smoothed quadratic energies on meshes. *ACM Trans. Graph. 34,* 1, 2:1–2:12.

MULLEN, P., TONG, Y., ALLIEZ, P., AND DESBRUN, M. 2008. Spectral conformal parameterization. In *Proc. Symposium on Geometry Processing*. 1487–1494.

MYLES, A., PIETRONI, N., AND ZORIN, D. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph. 33,* 4.

MYLES, A. AND ZORIN, D. 2012. Global parametrization by incremental flattening. *ACM Trans. Graph. 31,* 4, 109:1–109:11.

NOCEDAL, J. AND WRIGHT, S. J. 2006. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Berlin. NEOS guide http://www-fp.mcs.anl.gov/otc/Guide/.

PIGHIN, F. AND LEWIS, J. 2007. Practical least-squares for computer graphics. In *ACM SIGGRAPH 2007 Courses*.

PORANNE, R., OVREIU, E., AND GOTSMAN, C. 2013. Interactive planarization and optimization of 3D meshes. *Comput. Graph. Forum 32,* 1, 152–163.

RABINOVICH, M. 2016. Scalable locally injective mappings. `https://github.com/MichaelRabinovich/Scalable-Locally-Injective-Mappings`.

SANAN, P. D. 2014. Geometric elasticity for graphics, simulation, and computation. Ph.D. thesis, California Institute of Technology.

SCHENK, O., BOLLHÖFER, M., AND RÖMER, R. A. 2008. On large-scale diagonalization techniques for the Anderson model of localization. *SIAM Rev. 50,* 1, 91–112.

SCHENK, O., WCHTER, A., AND HAGEMANN, M. 2007. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Computational Optimization and Applications 36,* 2-3, 321–341.

SCHREINER, J., ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. 2004. Inter-surface mapping. *ACM Trans. Graph. 23,* 3.

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. *Computer Graphics Forum 32,* 5, 125–135.

SHEFFER, A. AND DE STURLER, E. 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers 17,* 3, 326–337.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. ABF++: Fast and robust angle based flattening. *ACM Trans. Graph. 24,* 2, 311–330.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Found. Trends Comput. Graph. Vis. 2,* 2, 105–171.

SMITH, J. AND SCHAEFER, S. 2015. Bijective parameterization with free boundaries. *ACM Trans. Graph. 34,* 4, 70:1–70:9.

SORKINE, O. AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proc. Symp. Geometry Processing*. 109–116.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '05. ACM, New York, NY, USA, 181–190.

TUTTE, W. T. 1963. How to draw a graph. *Proceedings of the London Mathematical Society 13*, 743–767.

WEBER, O., MYLES, A., AND ZORIN, D. 2012. Computing extremal quasiconformal maps. *Comput. Graph. Forum 31,* 5.

YOSHIZAWA, S., BELYAEV, A., AND SEIDEL, H.-P. 2004. A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of the Shape Modeling International 2004*. SMI '04. IEEE Computer Society, Washington, DC, USA, 200–208.

ZAYER, R., LÉVY, B., AND SEIDEL, H.-P. 2007. Linear angle based parameterization. In *Proc. SGP*. 135–141.

APPENDIX

A. SOLVING EQ. (21)

In order to solve Eq. (21), we write it in matrix form, and in terms of the coordinates $\mathbf{x}$. Then Eq. (21) is transformed into

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \tag{41}$$

where the structure of $\mathbf{A}$ and $\mathbf{b}$ in the 2D case is as follows. Assuming a set of orthogonal frames per element are prescribed, we let D_x, D_y be the FE gradient matrices of the mesh w.r.t. the frames. Additionally, we define four diagonal matrices, $\mathbf{W}_{ij}$ for $i, j = 1, 2$, where the diagonal of $\mathbf{W}_{ij}$ holds the (i, j) entries of all of the weights $\mathbf{W}_f$. In other words, $\mathbf{W}_{ij} = \operatorname{diag}(\{\mathbf{W}_f(i, j)\}_f)$. Similarly, we define $\mathbf{R}_{ij}$ to be the column vector holding the (i, j)

entries of all of the $\mathbf{R}_f$

$$\mathbf{A} = \mathbf{W}\mathbf{D} = \begin{pmatrix} \mathbf{W}_{11} & 0 & \mathbf{W}_{12} & 0 \\ \mathbf{W}_{21} & 0 & \mathbf{W}_{22} & 0 \\ 0 & \mathbf{W}_{11} & 0 & \mathbf{W}_{12} \\ 0 & \mathbf{W}_{21} & 0 & \mathbf{W}_{22} \end{pmatrix} \begin{pmatrix} D_x & 0 \\ D_y & 0 \\ 0 & D_x \\ 0 & D_y \end{pmatrix} \tag{42}$$

$$\mathbf{b} = \begin{pmatrix} \mathbf{R}_{11} \\ \mathbf{R}_{21} \\ \mathbf{R}_{12} \\ \mathbf{R}_{22} \end{pmatrix} \tag{43}$$

This can be readily solved by any least squares minimization algorithm.

B. SECTION 3 PROOFS

B.1 Lemma 3.1

Lemma 3.1 *Let* $\mathbf{J} = \mathbf{U}\mathbf{S_J}\mathbf{V}^\top$ *be the Singular Value Decomposition of* $\mathbf{J}$. *Then,*

$$\mathcal{D}(\mathbf{J}) = \mathcal{D}(\mathbf{S_J}) \tag{44}$$

$$\nabla_\mathbf{J}\mathcal{D}(\mathbf{J}) = \mathbf{U}\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})\mathbf{V}^\top \tag{45}$$

PROOF. Eq. (44) is immediate from the definition in Eq. (1). As for Eq. (45), we use the formula for the derivative of the singular values (see [Giles 2008])

$$\nabla_\mathbf{J}\mathcal{D}(\mathbf{S_J}) = \mathbf{U}\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J})\mathbf{V}^\top \tag{46}$$

□

B.2 Local step, general construction: proof of Eq. (39)

For a rotation invariant $\mathcal{D}(\mathbf{J})$ that is separably strictly convex in singular values, Eq. (38) can be satisfied by setting $(\mathbf{S}_\Lambda)_i$ such that:

$$\frac{\partial}{\partial\sigma_i}\mathcal{D}_{(\sigma_1,\ldots,\sigma_{i-1},\sigma_{i+1},\ldots,\sigma_d)}(\sigma_i) = 0 \tag{47}$$

In particular, in the case of a true isometric distortion measure, (38) is satisfied by setting the local step as the closest rotation $\Lambda = \mathbf{U}\mathbf{V}^\top$.

This can be seen from the fact that every partial function of $\mathcal{D}(\sigma)$ is strictly convex on $\mathbb{R}_{>0}$, and therefore has a single minimum, $(\mathbf{S}_\Lambda)_i$. Hence, for every $\sigma_i < (\mathbf{S}_\Lambda)_i$, $\frac{\partial}{\partial\sigma_i}\mathcal{D}_{(\sigma_1,\ldots,\sigma_{i-1},\sigma_{i+1},\ldots,\sigma_d)}(\sigma_i) < 0$, and for every $\sigma_i > (\mathbf{S}_\Lambda)_i$, $\frac{\partial}{\partial\sigma_i}\mathcal{D}_{(\sigma_1,\ldots,\sigma_{i-1},\sigma_{i+1},\ldots,\sigma_d)}(\sigma_i) > 0$. This is also true for $(\mathbf{S_J} - \mathbf{S}_\Lambda)_i$ and so Eq. (39) is satisfied.

B.3 Conformal energy local step derivation

Let $\mathcal{D}(\mathbf{J}) = \frac{\mathrm{tr}(\mathbf{J}^\top\mathbf{J})}{\det(\mathbf{J})^{2/d}}$. $\mathcal{D}(\mathbf{J})$ is rotation invariant and can be written as $\mathcal{D}(\sigma) = \frac{\sum_{i=1}^d \sigma_i^2}{\sigma_1\ldots\sigma_d}$. By differentiating the distortion measure w.r.t. the singular values in the 2D case, we find that

$$(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_1 = \frac{1}{\sigma_2} - \frac{\sigma_2}{\sigma_1^2},$$

and similarly, $(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_2 = \frac{1}{\sigma_1} - \frac{\sigma_1}{\sigma_2^2}$. Assuming $\mathbf{J}$ is *not* a similarity already, then, since $\sigma_1 > \sigma_2 > 0$, the first entry is negative, while the second is positive. By choosing $\sigma_1 > (\mathbf{S}_\Lambda)_i > (\sigma_2)$, this holds true for $(\mathbf{S_J} - \mathbf{S}_\Lambda)_i$ and so Eq. (38) is satisfied.

For the 3D case,

$$(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_i = \frac{-2\sigma_{i+1}\sigma_{i+2}(\sigma_{i+1}^2 + \sigma_{i+2}^2 - 2\sigma_i^2)}{(3\sigma_i\sigma_{i+1}\sigma_{i+2})^{5/3}},$$

where the index i cycles from 1 to 3 (i.e., $\sigma_4 = \sigma_1$). This is zero only for $\sigma_i = \sqrt{\frac{\sigma_{i+1}^2+\sigma_{i+2}^2}{2}}$, and so $(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_1 < 0$, $(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_3 > 0$. We note that $\bar{\sigma} = \sqrt{\frac{\sigma_1^2+\sigma_3^2}{2}}$ satisfies $\sigma_3 < \bar{\sigma} < \sigma_1$. Therefore, by choosing $\mathbf{S}_\Lambda = \bar{\sigma}\mathbf{U}\mathbf{V}^\top$, we get $(\mathbf{S_J} - \mathbf{S}_\Lambda)_1 < 0$, $(\mathbf{S_J} - \mathbf{S}_\Lambda)_3 > 0$, and by construction, same as the proof for Eq. (39), we get that the sign of $(\mathbf{S_J} - \mathbf{S}_\Lambda)_2$ is equal to the sign of $(\nabla_{\mathbf{S_J}}\mathcal{D}(\mathbf{S_J}))_2$.

C. CONNECTION WITH NEWTON'S METHOD

We show that our proxy can be written in a "Newton" form. Minimizing the proxy energy (21), generated with our algorithm for a rotation invariant $\mathcal{D}(\mathbf{J})$, provides a search direction that satisfies:

$$\mathbf{d^k} = \mathbf{p}^k - \mathbf{x}^{k-1} = -L_w^{-1}\mathbf{g}, \tag{48}$$

where L_w is the l.h.s. of Eq. (42) and g is the gradient of the energy at the given point. We note that, the L_w matrix is dependent on the weights, and thus changes at every iteration. Replacing it with the Hessian of the energy is exactly Newton's method, while replacing it with a constant Laplacian L is the approach of [Kovalsky et al. 2016]. The latter coincides with our approach and the original local/global approach of [Liu et al. 2008] only when used to minimize the ARAP energy. To see that (48) holds, note that the matching gradient condition (12) for our proxy distortion implies:

$$\nabla_{\mathbf{J}_f}\mathcal{D}(\mathbf{J}_f) = \mathbf{W}_f^{k\top}\mathbf{W}_f^k(\mathbf{J}_f - \Lambda_f), \tag{49}$$

where $\mathbf{W}_f^k$ are per-face matrix weights satisfying Eq. (12). Using the chain rule we get the energy gradient as a function of the vertices; in the notation of Eq. (42):

$$\mathbf{g} = D^\top\mathbf{W}^\top\mathbf{W}D(\mathbf{J} - \Lambda), \tag{50}$$

where $\mathbf{J}$ are the vectorized Jacobians, and $\Lambda = \mathbf{b}$ is the same as in Eq. (43).