SIAM J. Imaging Sciences
Vol. 15, No. 2, pp. 925–959

A Splitting Scheme for Flip-Free Distortion Energies*

Oded Stein†, Jiajin Li‡, and Justin Solomon†

Abstract. We introduce a robust optimization method for flip-free distortion energies used, for example, in parametrization, deformation, and volume correspondence. This method can minimize a variety of distortion energies, such as the symmetric Dirichlet energy and our new symmetric gradient energy. We identify and exploit the special structure of distortion energies to employ an operator splitting technique, leading us to propose a novel alternating direction method of multipliers (ADMM) algorithm to deal with the nonconvex, nonsmooth nature of distortion energies. The scheme results in an efficient method where the global step involves a single matrix multiplication and the local steps are closed-form per-triangle/per-tetrahedron expressions that are highly parallelizable. The resulting general-purpose optimization algorithm exhibits robustness to flipped triangles and tetrahedra in initial data as well as during the optimization. We establish the convergence of our proposed algorithm under certain conditions and demonstrate applications to parametrization, deformation, and volume correspondence.

Key words. computer graphics, optimization, nonconvex optimization, parametrization, ADMM

AMS subject classifications. 65K10, 90C26, 65D18, 68U05

DOI. 10.1137/21M1433058

1. Introduction.

Distortion energies measure how much a mapping from one shape to another deforms the initial shape. Minimizing these energies can yield maps between domains with as little distortion as possible. Minimization of distortion energies with a variety of constraints is employed in a wide array of computer graphics applications, such as UV mapping (where one seeks to embed a 3D surface into 2D while minimizing distortion, Figure 1, left), deformation (where parts of a surface or volume are deformed, and the goal is to find the overall deformation with least distortion, Figure 1, center), and volume correspondence (two boundary surfaces are given, and a distortion-minimizing map between the two volumes is desired, Figure 1, right). We are interested in computing maps that minimize distortion energies on triangle and tetrahedral meshes.

Flip-free distortion energies comprise an important subset of distortion energies. A mapping that minimizes such an energy will never invert (flip) a triangle or tetrahedron. Flip-free

*Received by the editors July 12, 2021; accepted for publication (in revised form) November 15, 2021; published electronically June 28, 2022.
https://doi.org/10.1137/21M1433058
Funding: The work of the authors was supported by the Swiss National Science Foundation's Early Postdoc.Mobility fellowship, the Army Research Office grant W911NF2010168, other Air Force Offiice of Scientific Research grant FA9550-19-1-031, the National Science Foundation grant IIS-1838071, the CSAIL Systems that Learn program, the MIT-IBM Watson AI Laboratory, the Toyota-CSAIL Joint Research Center, a gift from Adobe Systems, the MIT.nano Immersion Lab/NCSOFT Gaming Program seed grant, and the Skoltech-MIT Next Generation Program.

†Massachusetts Institute of Technology, Cambridge, MA USA (ostein@mit.edu, jsolomon@mit.edu).
‡The Chinese University of Hong Kong, Hong Kong (jjli@se.cuhk.edu.hk).

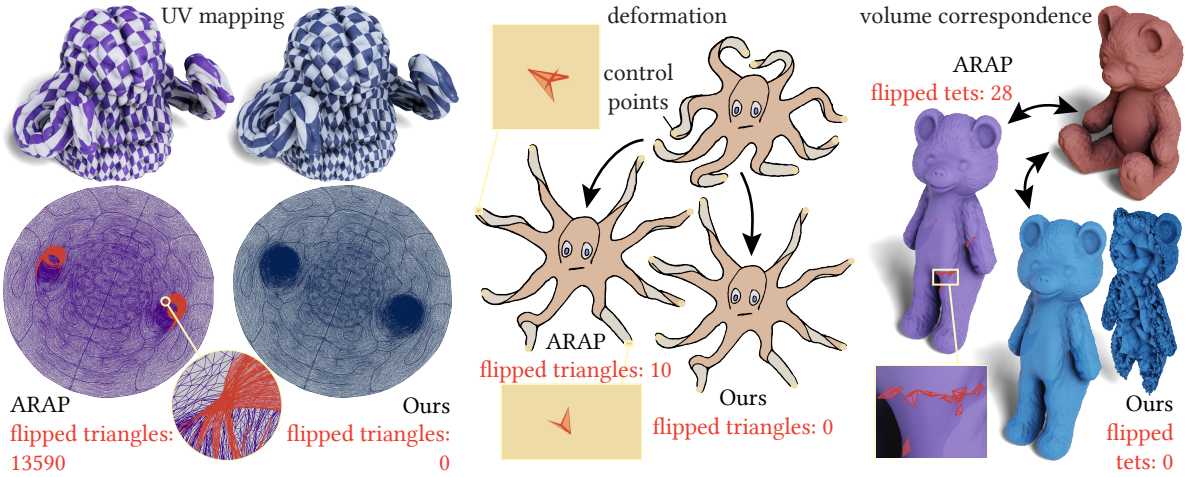


Figure 1. *Minimizing distortion energies in a variety of applications using our splitting method: UV mapping* (left, computing a distortion-minimizing map from the surface to $\mathbb{R}^2$)*, shape deformation* (center, fixing control points to deformed position and finding the distortion-minimizing map)*, volume correspondence* (right, finding the distortion-minimizing map between the interior of two different surfaces)*. Our method produces a flip-free result, unlike methods based on energies such as ARAP (E_{A}), which can exhibit flips when performing the same operation (flipped elements in red).*

distortion energies are difficult to optimize: they are usually nonlinear and nonconvex, and have singularities that correspond to collapsed elements. Typical optimization methods based on line-search require feasible, flip-free iterates. Thus, they must exercise great care to avoid singularities, where they will fail. For applications such as volume correspondence, it is difficult to even initialize with a feasible flip-free configuration. Certain line-search-free approaches can struggle with the problem's nonconvexity in both the objective function and the feasible set.

We focus on distortion energies that depend only on the mapping's Jacobian, are invariant to rotations, and are convex over symmetric positive definite matrices. This includes popular flip-free distortion energies such as the *symmetric Dirichlet energy*, as well as our new *symmetric gradient energy*. Previous methods optimizing the symmetric Dirichlet energy can, to our knowledge, not be mathematically proven to converge in the limit. We exploit the convexity in these energies by splitting the Jacobian of the mapping W into a rotational part U and a flip-free, symmetric part P. We propose a novel alternating direction method of multipliers (ADMM) algorithm to leverage this splitting in an efficient fashion, which results in three subproblems: optimize the vertex positions of the mapping W, optimize the Jacobian's rotational part U, and optimize the Jacobian's rotation-free part P. The optimization in W is linear, the optimization in U is an explicitly solvable Procrustes problem, and the optimization in P (the only part containing the objective function) has a closed-form solution for both energies considered in this article. The optimizations in U and P also decouple over triangles/tetrahedra, and can thus be parallelized, while the optimizations in W and P are convex.

In our approach, the target mapping does not need to be flip-free for every iteration until convergence is attained; since P is always flip-free, the distortion energy will never be singular and can be evaluated even if W contains flips. Thus, our approach is naturally robust to flipped elements in the iterates and can be initialized with flipped elements.

ADMM-based algorithms are, in general, not guaranteed to converge for nonconvex nonlinear problems, such as the ones involving distortion energies. For our method, however, we can present a theoretical analysis of convergence behavior that can show convergence given certain conditions. This mathematical proof goes beyond what is usual for other flip-free optimization algorithms, yielding the first optimization of the symmetric Dirichlet energy that can be proven to converge. Beyond describing the circumstances under which we reach a critical point of the optimization problem, this analysis significantly informs our algorithm: it lets us automatically choose appropriate augmented Lagrangian penalty weights.

Our contributions are

- a parallelizable optimization method for nonlinear nonconvex flip-free distortion energies that is robust to the presence of flipped triangles in the initial data;
- a convergence analysis that discusses the convergence of our algorithm to a stationary point given certain conditions;
- a novel distortion energy, the *symmetric gradient energy*, which yields flip-free distortion-minimizing maps that are similar to popular non-flip-free methods (but are flip-free).

We demonstrate our method on applications in UV parametrization, surface and volume deformation, and volume correspondence (see Figure 1).

2. Related work.

2.1. Optimizing distortion energies. Distortion energies have a long history in geometry processing and related fields like physical simulation and differential geometry. They are part of tools for parametrization, deformation, and related tasks.

Early approaches include optimizing harmonic and conformal energies to produce angle-preserving mappings with a variety of optimization methods [26, 28, 37, 54, 67, 94]. As they can be measured and optimized efficiently, conformal energies remain popular and are the subject of ongoing research [35, 86, 93, 99, 103].

A different way to measure distortion is to quantify the deviation of a mapping's local structure from rotation, as in the as-rigid-as-possible (ARAP) energy [101], which can be efficiently optimized using a per-element local-global approach [60]. ARAP is similar to other quasi-elastic energies [19].

Optimization of flip-free distortion energies goes back to the work of Tutte [108], who showed that minimizing Tutte's energy while fixing boundary vertices to a convex polygon yields a flip-free mesh parametrization. Recent methods compute flip-free maps while simultaneously minimizing some kind of distortion for surfaces [53, 57], volumes [2], simplicial maps [58], nonstandard boundary conditions [112], or with a focus on numerical robustness [95]. Conformal and harmonic energies can be augmented to produce flip-free maps, e.g., by including cone singularities in the parametrization [39]. Cone singularities can be used in a variety of ways to produce parametrizations [20, 99].

The symmetric Dirichlet energy [91, 98] combines the idea of measuring the deviation of the Jacobian from the identity with flip-free maps: the energy is singular for zero-determinant Jacobians, which means that during the minimization process elements cannot collapse and invert. Since the symmetric Dirichlet energy is nonconvex and singular, it requires specialized optimization algorithms. The symmetric Dirichlet energy can be optimized in a wide variety of ways: with a modified line-search to avoid the energy's singularities in an L-BFGS-style

optimization (augmented with techniques for global bijectivity) [98], with a quadratic proxy to accelerate convergence [53], with a local-global modification of line-search that can also be applied to a variety of other rotation-independent distortion energies [84], with a preconditioned line-search based on Killing field approximation [22], and by progressively adjusting the reference mesh [59]. These approaches require initialization with a flip-free map whose distortion is then further reduced.

A different approach to generating flip-free distortion-minimizing relies on custom distortion energies that can be optimized efficiently without line-search methods [34]. Concurrent work introduces a framework for flip-free conformal maps [35], and a framework for globally elastic 3D deformations using physical simulation methods [30]. Yet other approaches include initializing with a flip-free map by lifting the mesh to a higher dimension and achieving injectivity that way [27], and applying block coordinate descent [70]. Using a barrier-aware line-search and quasi-Newton methods, one can optimize a variety of distortion energies [121]. One can also discretize physical elasticity energies that naturally model distortion and carry out a physical simulation [97].

The concurrent work WRAPD [17] also uses ADMM to compute flip-free distortion-minimizing maps, but with a different splitting technique than ours. They use two-block ADMM (compared to our three blocks with variables $\mathbf{W}$, $\mathbf{U}$, $\mathbf{P}$), where the nonconvex ADMM substep requires a line-search optimization to be solved to convergence every step (without guarantees on how this might interfere with the ADMM's convergence). They do not include a convergence proof (which we do).

There are many other approaches for efficient flip-free distortion minimization [3, 21, 32, 70, 104, 105, 116] For certain applications, such as quad meshing [51], surface-to-surface mapping [29, 89, 90], joint optimization of map and domain [56], globally bijective mapping [50], and input-aligned maps [69], distortion energies with special properties are used.

2.2. Alternating direction method of multipliers. The augmented Lagrangian method and the ADMM are popular optimization methods [16]. While the convergence of ADMM is well known for convex problems, convergence can also be proven in some other scenarios that often necessitate special proofs, such as classes of weakly convex problems [120], certain nonconvex ADMMs with *linear* constraints [43, 111, 118], nonconvex and nonlinear, but equality-constrained problems [110], and bilinear constraints [117]. For specific nonlinear nonconvex ADMMs, specialized proofs exist [33, 115]. Our method does not exactly fit any of the above approaches, but uses ideas from many of them to analyze convergence, such as an explicit boundedness condition [117], the use of a potential function [118], and the Kurdyka-Łojasiewicz condition [33].

ADMM has been employed in many computer graphics and image processing applications. It is especially useful when a convex problem can be split into multiple simpler subproblems that are each convex—in that case, convergence of the method follows from standard results [16, section 3.2]. Such convex ADMM is used, for example, to produce developable surfaces after convex relaxation [92], to speed up optimization in computer vision and machine learning [113], for aligning point sets with rigid transforms through relaxation of a nonconvex problem [85], as a substep in a rotation-strain simulation of elastic objects [79], to compute the earth mover's distance [100], and for isogeometric analysis after transforming a nonconvex problem into a biconvex problem [73].

When a problem is nonconvex, ADMM is more difficult to employ in a way that assures convergence. As a result, some applications of ADMM do not provide an explicit convergence guarantee but are able to show convergence empirically. Past work uses nonconvex ADMM with a linear constraint for the physical simulation of elastic bodies with collisions [75], an approach applied later to character deformation [66] and cloth simulation [65]. In later concurrent work this approach is extended to globally injective maps [76] by adding a step to the ADMM that promotes injectivity through a nonlinear optimization procedure while temporarily tolerating noninjective maps; this ADMM uses a line-search in the inner loop.

For some specific nonconvex applications of ADMM, convergence can be proven, just as we do in this article, although these examples do not cover our use case [74, 119].

Unlike past applications of ADMM to the problem of distortion-minimizing maps, our distortion minimization technique has all of the following features:

- We solve a nonconvex problem with a *nonlinear constraint*, $(G\mathbf{W})_i = \mathbf{U}_i\mathbf{P}_i$.
- Our splitting contains *three* ADMM blocks instead of the usual two, designed so that each block is solvable in closed form.
- We present a convergence analysis specialized to our algorithm; to our knowledge, this theoretical analysis is new and adds to the cases in which nonconvex multiblock ADMM is proven to converge.

3. Problem setup.

3.1. Preliminaries.

We compute a map from a *source mesh* to a deformed *target mesh* composed of triangle or tetrahedra. The goal is to measure the distortion of the map and to optimize the map (and the target mesh) subject to certain constraints. $\mathbf{V} \in \mathbb{R}^{n\times d_\iota}$ contains the coordinates of the vertices of the source mesh, where n is the number of vertices and $d_\iota = 2, 3$ is their dimension. The individual vertices are denoted by $\mathbf{V}_i, i = 1, \ldots, n$.

The optimization variable $\mathbf{W} \in \mathbb{R}^{n\times d_o}$ contains the target coordinates of the vertices under the map, where $d_o \leq d_i$ is the dimension of the output vertices. The dimension d_o can be different from the input dimension, for example, when we compute the UV mapping of a surface in 3D, where $d_\iota = 3$ and $d_o = 2$. The individual vertices are denoted by $\mathbf{W}_i, i = 1, \ldots, n$. The number of triangles or tetrahedra (elements) in the mesh is m and the dimension of the elements is d ($d = 2$ for triangles and $d = 3$ for tetrahedra). w_i is the area or volume of the ith element, depending on d. We will deal with collections of matrices associated with each triangle or tetrahedron of a mesh. For this purpose, we let $(\mathbb{R}^{d\times d})^m$ be the space of m independent $d \times d$ matrices. If $\mathbf{J} \in (\mathbb{R}^{d\times d})^m$, we denote by $\mathbf{J}_i$ the ith matrix in $\mathbf{J}$.

Our energies depend on the Jacobian of the map $\mathbf{V} \mapsto \mathbf{W}$. Assuming our map is affine when restricted to the interior of each element, the Jacobian is piecewise constant.

Definition 3.1 (piecewise constant Jacobian). *Let $\mathbf{V} \in \mathbb{R}^{n\times d_\iota}$ be the source coordinates and $\mathbf{W} \in \mathbb{R}^{n\times d_o}$ the target coordinates of a mapping of a mesh with m triangles or tetrahedra. Then $G : \mathbb{R}^{n\times d_o} \to (\mathbb{R}^{d_o\times d_o})^m$ is the linear operator (dependent on $\mathbf{V}$) such that the mapping's Jacobian for the ith triangle or tetrahedron is given by the matrix $(G\mathbf{W})_i$.*

Supplemental material describes how to compute the Jacobian from target vertex positions.

We use the Frobenius product and norm for vectors and matrices. The *Frobenius product* (or *dot product* for vectors) is defined as $X \cdot Y \coloneqq \operatorname{trace} X^\top Y$. The *Frobenius norm* (or L^2 norm for vectors) is defined via $\|X\|^2 \coloneqq X \cdot X$.

At last we define spaces for the rotation and flip-free parts of the Jacobian.

Definition 3.2 (rotation and semidefinite matrices). $\mathrm{SO}(d)$ *is the space of rotation matrices of dimension* d. $\mathcal{S}_+^d$ *is the space of symmetric positive definite (spd) matrices of dimension* d.

3.2. Optimization target. We can understand a variety of algorithms for parametrization, deformation, and related tasks as optimizing a generic energy of the following form:

$$E_{\text{generic}}(\mathbf{W}) \coloneqq \sum_{i=1}^{m} w_i f\big((G\mathbf{W})_i\big) \,. \tag{3.1}$$

Here, $f(\cdot)$ denotes a per-element distortion energy, summed over the triangles/tetrahedra i of the mesh; we call f the *defining function* of our problem. It is evaluated on the Jacobian $G\mathbf{W}$ of the map $\mathbf{V} \mapsto \mathbf{W}$ and is typically designed to be a rotation-invariant function quantifying how much $(G\mathbf{W})_i$ deviates from being a rigid motion. Rotation invariance implies we can think of $f(\cdot)$ as a function of the singular values of $(G\mathbf{W})_i$.

Each possible choice of distortion energy $f(\cdot)$ captures a different trade-off between different means of deforming the source domain onto the target, e.g., between angle and area preservation. Below, we discuss some choices for $f(\cdot)$ used in our experiments; we refer the reader to [84, Table I] for an exhaustive list, many of which can be easily plugged into our optimization framework. In section 7, we propose one additional option, the symmetric gradient energy, which appears to yield flip-free maps in practice that are similar to the popular—but not flip-free—ARAP energy (discussed in Appendix A.2.3).

Of particular interest are distortion energies that explicitly avoid inverted elements in the computed mapping. To achieve this, we augment (3.1) with a term that forbids flipped triangles or tetrahedra, and add conditions on f:

$$\boxed{E(\mathbf{W}) \coloneqq \sum_{i=1}^{m} w_i f\big((G\mathbf{W})_i\big) + \chi_+\big(\det(G\mathbf{W})_i\big) \,.} \tag{3.2}$$

Here, χ_+ denotes the indicator function

$$\chi_+(x) \coloneqq \begin{cases} 0 & \text{if } x \geq 0, \\ \infty & \text{otherwise.} \end{cases}$$

If f is smooth on all Jacobians with positive determinant, then the energy E is smooth on all maps with positive determinant. The extra term in (3.2) containing the characteristic function χ_+ can be understood as a constraint preserving local injectivity of the map in the interior of each element. This needs to be combined with an appropriate $f(x)$ which goes to ∞ as $x \to 0$, to create an appropriate barrier that ensures the region where χ_+ is infinite is never reached. This constraint is implicit in the design of many past algorithms [22, 59, 84, 98], but we choose to expose it explicitly as it will inform our design in section 4.

The main thrust of our research is to propose an efficient algorithm for optimizing energies of the form (3.2). Table 1 offers a quick overview over all energies that are used in this article.

Table 1

A table summarizing all deformation energies featured in this article. Energies from previous work are introduced in more detail in Appendix A; the symmetric gradient energy is introduced in section 7.

Energy	Symbol	Definition	Discussed in...	Properties
Tutte	E_{T}	$E_{\mathrm{T}}(\mathbf{W}) = \sum_{\text{edges } (i,j)} \frac{\|\mathbf{w}_i - \mathbf{w}_j\|^2}{\|\mathbf{v}_i - \mathbf{v}_j\|}$	Appendix A.2.1	flip-free in 2D (not 3D); linear; no initialization needed
Conformal	E_{C}	(3.1) with $f(X) = \frac{1}{2}\|X\|^2$ plus additional area term	Appendix A.2.2	not flip-free; linear; no initialization needed
ARAP	E_{A}	(3.1) with $f(X) = f_{\mathrm{A}}(X) = \frac{1}{2}\|X - \operatorname{rot} X\|^2$, $\operatorname{rot} X$ is the rotational part of X	Appendix A.2.3	not flip-free; nonlinear; desirable rigidity property; needs initializer; efficient optimizer
Symmetric Dirichlet	E_{D}	(3.2) with $f(X) = f_{\mathrm{D}}(X) = \frac{1}{2}\left(\|X\|^2 + \|X^{-1}\|^2\right)$	Appendix A.1	flip-free; strong singularity; nonlinear; needs initializer
Symmetric gradient	E_{G}	(3.2) with $f(X) = f_{\mathrm{G}}(X) = \frac{1}{2}\|X\|^2 - \log\det X$	section 7	flip-free; weak singularity; nonlinear; needs initializer

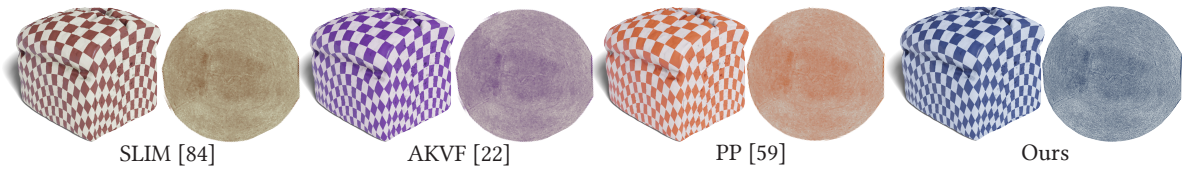


Figure 2. *Our optimization method visually matches the results of a variety of other methods when tasked with producing a UV map with the same energy E_{D}.*

Figure 2 shows our optimization reproducing a parametrization with an energy popular in previous work, the symmetric Dirichlet energy E_{D}.

4. Our optimization method. Our optimization method can be applied to distortion energies of the form (3.2) where the defining function f is convex over the set of symmetric positive semidefinite matrices $\mathcal{S}_+^d$ (a property which holds for many distortion energies). There are two main challenges in optimizing energies of this form. One challenge is the nonconvexity of the defining function f when applied to arbitrary matrices $(G\mathbf{W})_i \in \mathbb{R}^{d\times d}$. Another challenge is the nonsmoothness of the characteristic function χ_+ and potential singularities in f. Previous work solves these issues by, e.g., employing line-search methods which can handle nonconvex problems, and are specifically constructed to avoid the singular regions of χ_+ and f during time stepping (see the discussion in section 2).

We address these challenges differently, starting with the polar decomposition [38, Theorem 2.17]. Every matrix $J \in \mathbb{R}^{d\times d}$ with positive determinant can be decomposed into a rotation matrix $U \in \mathrm{SO}(d)$ and a symmetric matrix $P \in \mathcal{S}_+^d$ such that

$$J = UP\,. \tag{4.1}$$

Applying the decomposition to our problem, we can reformulate (3.2) as

$$\begin{aligned} \min_{\substack{\mathbf{W} \\ \mathbf{U}\in(\mathrm{SO}(d))^m \\ \mathbf{P}\in(\mathcal{S}_+^d)^m}} \quad & \sum_{i=1}^m w_i f(\mathbf{P}_i) \\ \text{s.t.} \quad & (G\mathbf{W})_i - \mathbf{U}_i\mathbf{P}_i = 0 \quad \forall i \,. \end{aligned} \tag{4.2}$$

This new formulation is now convex in both $\mathbf{W}$ and $\mathbf{P}$ if f is convex over the positive semi-definite cone $\mathcal{S}_+^d$: f_{G} and f_{D} are nonconvex for arbitrary matrices, but convex for matrices in $\mathcal{S}_+^n$, like many distortion energies. The nonconvexity is now entirely contained in $\mathbf{U}$. We have thus extracted the *hidden convexity of the problem*, and restricted the nonconvexity to rotational matrices only (this extraction of salient parts of the energy mirrors approaches that extract the singular values [22, 59, 84, 98] and appears in concurrent work [17]).

The constraint from (4.2) is still difficult to accommodate. We deal with this problem by employing an ADMM [16] tailored for our problem. Let $\mathbf{W}$ be our target vertex positions, and let $\mathbf{U} \in (\mathrm{SO}(d))^m$, $\mathbf{P} \in (\mathcal{S}_+^d)^m$, $\mathbf{\Lambda} \in (\mathbb{R}^{d\times d})^m$. Our *augmented Lagrangian function* is

$$\boxed{\Phi(\mathbf{W},\mathbf{U},\mathbf{P},\mathbf{\Lambda}) \coloneqq \sum_{i=1}^m w_i f(\mathbf{P}_i) + \frac{\mu_i}{2}\left(\|(G\mathbf{W})_i - \mathbf{U}_i\mathbf{P}_i + \mathbf{\Lambda}_i\|^2 - \|\mathbf{\Lambda}_i\|^2\right),} \tag{4.3}$$

where $\mu_i > 0$ are a set of m Lagrangian penalty weights. The variable $\mathbf{\Lambda}$ is a scaled Lagrange multiplier for the constraint $(G\mathbf{W})_i = \mathbf{U}_i\mathbf{P}_i$.

In the formulation of (4.3), the function f that contains a singularity for matrices with zero determinant is only ever evaluated on $\mathbf{P}_i$, which *cannot* have zero determinant, as it is symmetric and positive definite. If the Jacobian map $(G\mathbf{W})_i$ inverts or degenerates a triangle, i.e., has zero or negative determinant, this manifests as a feasibility error (which is finite).

The augmented Lagrangian method now consists of successively optimizing Φ in each of its primal arguments in an alternative way, and then updating the dual variable Λ. Our method is described in pseudocode form in Algorithm 4.1; a description of each of the substeps follows.

Algorithm 4.1. Three-block ADMM for flip-free distortion energies.

method SplittingOptimization $\left(\mathbf{W}^{(0)},\mathbf{U}^{(0)},\mathbf{P}^{(0)},\mathbf{\Lambda}^{(0)}\right)$:
for $k \leftarrow 1,\ldots$ **do**
  $\mathbf{W}^{(k)} \leftarrow \operatorname{argmin}_{\mathbf{W},A\mathbf{W}=b}\ \Phi\left(\mathbf{W},\mathbf{U}^{(k-1)},\mathbf{P}^{(k-1)},\mathbf{\Lambda}^{(k-1)}\right)$
  $\mathbf{U}^{(k)} \leftarrow \operatorname{argmin}_{\mathbf{U}}\ \Phi\left(\mathbf{W}^{(k)},\mathbf{U},\mathbf{P}^{(k-1)},\mathbf{\Lambda}^{(k-1)}\right) + p\left(\mathbf{U},\mathbf{U}^{(k-1)}\right)$
  $\mathbf{P}^{(k)} \leftarrow \operatorname{argmin}_{\mathbf{P}}\ \Phi\left(\mathbf{W}^{(k)},\mathbf{U}^{(k)},\mathbf{P},\mathbf{\Lambda}^{(k-1)}\right)$
  $\mathbf{\Lambda}_i^{(k)} \leftarrow \mathbf{\Lambda}_i^{(k-1)} + (G\mathbf{W}^{(k)})_i - \mathbf{U}_i^{(k)}\mathbf{P}_i^{(k)} \quad \forall i$

Updating W. To update $\mathbf{W}$ we optimize Φ with respect to $\mathbf{W}$. Φ is a quadratic function in $\mathbf{W}$, and can thus be optimized by solving the linear system

$$L\mathbf{W} = G^\top r \,, \tag{4.4}$$

where

$$r_i = \mu_i \left(\mathbf{U}_i \mathbf{P}_i - \Lambda_i\right) \in (\mathbb{R}^{d\times d})^m \qquad \text{and} \qquad L = \sum_{i=1}^{m} \mu_i G_i^\top G_i \in \mathbb{R}^{n\times n} .$$

As G is implemented as a simple finite element gradient matrix (see supplemental file M143305_01.pdf [local/web 217 KB]), we can write $L = G^\top M G$, where M is a mass matrix with the respective entries of μ_i on the diagonal. As a result, it is a sparse Laplacian matrix similar to the cotangent Laplacian [81].

At this step, we can also enforce constraints, e.g., for deformation, on the vertex positions $\mathbf{W}$. The quadratic optimization problem can be solved, with any feasible linear constraint of the form $A\mathbf{W} = b$, at negligible additional cost. In fact, since G maps constant functions to 0, there needs to be a minimum number of constraints to make (4.4) solvable; in the absence of any constraints (such as in UV mapping) we simply fix the first vertex of $\mathbf{W}$ to the origin.

Updating U. To update $\mathbf{U}$, we optimize Φ augmented with a proximal function p,

$$p(\mathbf{U}, \mathbf{U}^{(k-1)}) \coloneqq \sum_{i=1}^{m} \frac{h_i}{2} \left\| \mathbf{U}_i - \mathbf{U}_i^{(k-1)} \right\|^2 , \tag{4.5}$$

where $h_i > 0$ is the proximal parameter and $\mathbf{U}^{(k-1)}$ is the iterate from a previous step. This proximal function p is needed for the algorithm to converge (see section 5).

Both Φ and p decouple in $\mathbf{U}$ over elements, giving the problem

$$\underset{\mathbf{U}_i \in \mathrm{SO}(d)}{\operatorname{argmin}} \frac{\mu_i}{2} \left\| (G\mathbf{W})_i - \mathbf{U}_i \mathbf{P}_i + \mathbf{\Lambda}_i \right\|^2 + \frac{h_i}{2} \left\| \mathbf{U}_i - \mathbf{U}_i^{(k-1)} \right\|^2 . \tag{4.6}$$

Since $\mathbf{U}_i \in \mathrm{SO}(d)$ and $\mathbf{P}_i \in \mathcal{S}_+^d$, we get that (4.6) is equivalent to the Procrustes problem [36]

$$\underset{\mathbf{U}_i \in \mathrm{SO}(d)}{\operatorname{argmin}} \left\| \mathbf{U}_i - \mathbf{Q}_i \right\|^2 , \tag{4.7}$$

where $\mathbf{Q}_i = \left((G\mathbf{W})_i + \mathbf{\Lambda}_i\right) \mathbf{P}_i + \frac{h_i}{\mu_i} \mathbf{U}_i^{(k-1)}$. Supplemental material describes our approach to solving the Procrustes problem in detail; there is an explicit closed-form solution.

Updating P. To update $\mathbf{P}$ we optimize Φ with respect to $\mathbf{P}$. As Φ decouples in $\mathbf{P}$ over elements, we can optimize Φ separately for each triangle/tetrahedron,

$$\underset{\mathbf{P}_i \in \mathcal{S}_+^d}{\operatorname{argmin}} \; w_i f(\mathbf{P}_i) + \frac{\mu_i}{2} \left\| (G\mathbf{W})_i - \mathbf{U}_i \mathbf{P}_i + \mathbf{\Lambda}_i \right\|^2 . \tag{4.8}$$

Since both f_{G} and f_{D} are convex over $\mathcal{S}_+^d$, the problem (4.8) is convex. We merely need to find the single critical point by finding the solution in $\mathcal{S}_+^d$ of

$$w_i \nabla f(\mathbf{P}_i) + \mu_i \mathbf{P}_i = \mu_i \operatorname{symm}\left(\mathbf{U}_i^\top \left((G\mathbf{W})_i + \mathbf{\Lambda}_i \right) \right) , \tag{4.9}$$

where $\operatorname{symm}(\cdot)$ symmetrizes a matrix: $\operatorname{symm}(X) = \frac{1}{2}(X + X^\top)$. This can be solved explicitly in closed form for both f_{G} and f_{D}. Due to floating point issues, closed-form solvers for f_{D} can fail in certain scenarios; we then use a simple iterative scheme (see supplemental file M143305_01.pdf [local/web 217 KB]).

Updating Λ. To update the estimate of the Lagrange multiplier Λ, we apply a gradient ascent approach for the scaled augmented Lagrangian method [16, section 3.1.1],

$$\mathbf{\Lambda}_i = \mathbf{\Lambda}_i^{(k-1)} + (G\mathbf{W})_i - \mathbf{U}_i\mathbf{P}_i \,, \tag{4.10}$$

where $\mathbf{\Lambda}^{(k-1)}$ is the iterate from a previous step of the optimization method.

As we will see in section 5, Algorithm 4.1 can be proven to converge under certain conditions. The algorithm requires the choice of both a Lagrangian penalty parameter μ_i, as well as a proximal parameter h_i. The choice of both of these will be informed directly by the proof. The actual algorithm implemented in our code (which is slightly different), the initialization of $\mathbf{W}^{(0)}, \mathbf{U}^{(0)}, \mathbf{P}^{(0)}, \mathbf{\Lambda}^{(0)}$, as well as the termination condition, are discussed in section 6.

Robustness to flipped elements. Algorithm 4.1 can be robust with respect to flipped triangles in the target mesh iterate $\mathbf{W}$. Since the defining energy function f is only ever evaluated on the iterate $\mathbf{P}$, which consists of symmetric positive definite matrices ($\mathcal{S}_+^d$) this evaluation can never be undefined, *even if* the target mesh iterate $\mathbf{W}$ currently contains flipped triangles. This is a property of the augmented Lagrangian method's weakly enforced constraint. $(G\mathbf{W})_i = \mathbf{U}_i\mathbf{P}_i$ is only ever strongly active when the algorithm has converged, allowing us to circumvent the problem of evaluating f of a singularity which can affect previous work based on line-search optimization. We can use this robustness property to unflip parametrizations produced by other methods that do not guarantee flip-free minimizers, such as E_{A}. In Figure 3 we initialize our optimization with the result of an E_{A} UV map, and run it until the map contains no flipped triangles, unflipping the triangles left over by E_{A}. The limits of this robustness are discussed in section 9.

Closed-form solutions for every substep. The four substeps of our method are explicitly computable with closed-form solutions for the energies E_{G} and E_{D} (unlike, e.g., [17]) and do not require parameter tuning. This makes our method easy to implement.

Efficient evaluation and parallelization. Every step of our method can be computed efficiently. The $\mathbf{W}$ step contains only a single linear solve, and the matrix L does not change as long as the penalties μ_i do not change. This allows us to precompute the decomposition once, and only apply a cheap backsubstitution every iteration. To achieve this, we use Suitesparse's CHOLMOD if the constraints make the problem in $\mathbf{W}$ definite, and Suitesparse's UMFPACK if the constraints result in an indefinite problem [25]. The $\mathbf{U}$, $\mathbf{P}$ and $\mathbf{\Lambda}$ steps all decouple

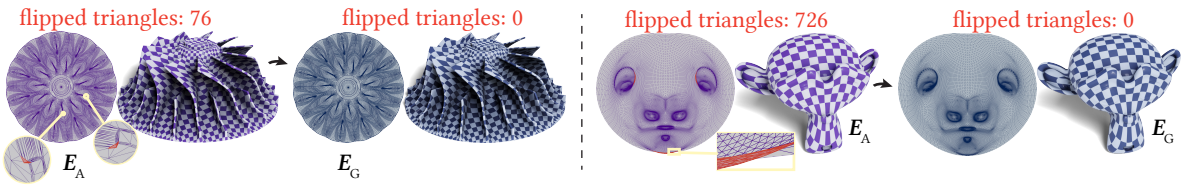


Figure 3. *Using the robustness of our method with respect to flipped triangles in the initial iterate, we can use our method to unflip the outputs of other methods. In this example, a flip-containing UV parametrization computed with E_{A} is unflipped by running our method to optimize E_{G} until a flip-free configuration is obtained, resulting in a parametrization that is very similar to E_{A}'s, but flip-free (flipped triangles in red).*

over elements, and can thus be computed for each triangle/tetrahedron separately, in parallel. This makes the algorithm highly parallelizable. We implement parallelization using OpenMP.

5. Convergence analysis. Even though the proposed Algorithm 4.1 looks like the usual ADMM scheme, there is an important caveat: the constraints in our formulation (4.2) are *nonlinear* and *nonconvex*. As a result, standard ADMM convergence analysis [16] does not apply. This distinguishes our method from many other computer graphics ADMM methods discussed in section 2.2. As (4.2) is nonconvex, a natural question is whether the ADMM will converge or not. We confirm that this is true in this section. Specifically, we show that, under certain conditions (which in practice often hold), the sequence $\{(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})\}_{k\geq 0}$ generated by the proposed ADMM method will converge to a Karush–Kuhn–Tucker (KKT) point $(\mathbf{W}^*, \mathbf{U}^*, \mathbf{P}^*, \mathbf{\Lambda}^*)$ of (4.2) that is defined by the conditions

$$\begin{cases} w_i \nabla f(\mathbf{P}_i^*) - \mu_i {\mathbf{U}_i^*}^\top \mathbf{\Lambda}_i^* = 0 \quad \forall i, \\ \partial g(\mathbf{U}_i{}^*) - \mu_i \mathbf{\Lambda}_i^* {\mathbf{P}_i^*}^\top \ni 0 \quad \forall i, \\ (G\mathbf{W}^*)_i = \mathbf{U}_i{}^* \mathbf{P}_i{}^* \quad \forall i. \end{cases} \tag{5.1}$$

Here, $g(\mathbf{U}_i)$ is the indicator function over the rotation matrix set $\mathrm{SO}(d)$ and $\partial g(\cdot)$ is the limiting subdifferential. We refer to [55] for a recent review of stationarity in nonconvex problems for the concrete definition. The second condition of the KKT system can be written as $\mathbf{U}_i{}^* = \mathrm{argmin}_{\mathbf{U}_i \in \mathrm{SO}(d)} (\mathbf{U}_i \cdot \mathbf{\Lambda}_i^* {\mathbf{P}_i^*}^\top)$.

We now present the convergence condition. First, we introduce an explicit boundedness condition on the gradient norm of $f(\cdot)$ with respect to the sequence $\{\mathbf{P}^{(k)}\}$.

Condition 5.1. *The gradient of f evaluated at the iterates $(\mathbf{P}^{(k)})_{k\geq 0}$ generated by our ADMM algorithm is bounded, i.e., $\|\nabla f(\mathbf{P}_i^{(k)})\| \leq B_i \quad \forall i$.*

This condition must be verified by the user when employing the method. Such boundedness conditions are common for nonconvex ADMM [33, 117, 120]. Figure 4 shows that this bound holds in practice for a variety of UV parametrization experiments, but it does not have to hold for every problem (see section 9). This is because the energy function f is not globally, but only locally Lipschitz continuous. However, the usual convergence results for general nonconvex ADMM (e.g., even with linear constraints) rely on a global Lipschitz condition, which is the key for obtaining a bounded sequence for both primal and dual variables. Without global Lipschitz continuity it is challenging to bound the difference between two iterates. To address this, we build our convergence analysis on Condition 5.1, which not only enables us to use a local Lipschitz property, but also plays an important role in providing us with an explicit bound for the penalty parameter. The bound is crucial for our practical implementation as well, as elaborated in section 6.

Our second condition lets us bound the difference between $\Lambda_i^{(k+1)}$ and $\Lambda_i^{(k)}$.

Condition 5.2. *There exists a $\gamma > 0$ such that*

$$\left\|\Lambda_i^{(k+1)} - \Lambda_i^{(k)}\right\|^2 \leq \frac{\gamma}{4}\left\|\Lambda_i^{(k+1)} + U_i^{(k+1)} \left(\Lambda_i^{(k+1)}\right)^\top U_i^{(k+1)} - \Lambda_i^{(k)} - U_i^{(k)} \left(\Lambda_i^{(k)}\right)^\top U_i^{(k)}\right\|^2.$$

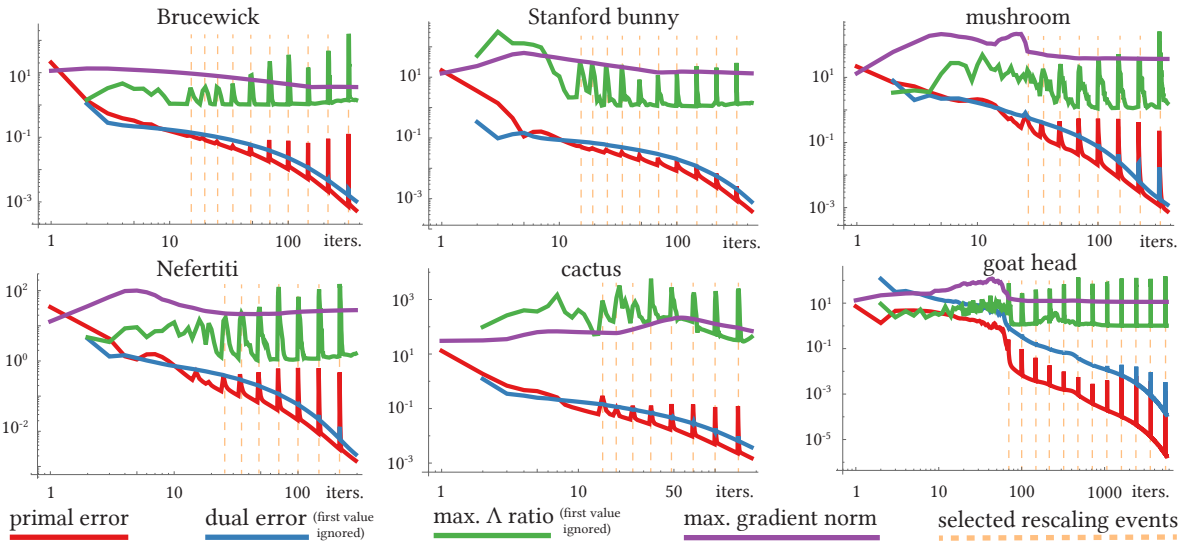


Figure 4. *Log-log plots of the optimizations performed in Figure 5 showing e^{prim}, e^{dual}, the largest energy gradient $\|\nabla f(\mathbf{P}_i^{(k)})\|$ (as a proxy for B_i from Condition 5.1, and the largest ratio $\|\Lambda_i^{(k+1)}-\Lambda_i^{(k)}\|/\|\frac{1}{2}(\Lambda_i^{(k+1)}-\Lambda_i^{(k)}+U_i^{(k+1)}(\Lambda_i^{(k+1)})^\top U_i^{(k+1)}-U_i^{(k)}(\Lambda_i^{(k)})^\top U_i^{(k)})\|$ (as a proxy for $\gamma^{1/2}$ from Condition 5.2, where the Λ_i are scaled by μ_i to be able to compare them across rescalings). The errors steadily decrease, and gradient and Λ ratio remain bounded, except for rescaling events that lead to temporary spikes that the bounds quickly recover from. Rescaling is not part of Algorithm 4.1 for which we analyze convergence, but are employed in Algorithm 6.1 to speed up the performance.*

This condition is needed to be able to bound the Λ update by the U and P updates. Because of the symmetrization in the P update (4.9), we lose information on the antisymmetric part of Λ—this additional condition allows us to control the antisymmetric part using only the symmetric part. Figure 4 shows appropriate bounds γ for a few UV parametrization applications. We conjecture that Condition 5.2 may not be necessary to prove convergence.

We leave the investigation of convergence behavior with weaker conditions to future work. These conditions are needed to enable our particular proof; different proof strategies could use other conditions. For many practical applications, however, our conditions are reasonable to impose. Figure 4 shows that these conditions are realistic for our UV parametrization examples presented in Figure 5. Both the gradient bound from Condition 5.1, as well as the Λ ratio from Condition 5.2 can be bounded during the optimization.

Since ADMM algorithms are primal-dual methods, the crux of our convergence analysis is to use the primal variable $(\mathbf{W}, \mathbf{U}, \mathbf{P})$ to bound the dual update of $\mathbf{\Lambda}$, leading to a sufficient decrease in the augmented Lagrangian function. To start, we derive an explicit local Lipschitz constant for various deformation energies, which is central to our convergence analysis.

Lemma 5.3. *We have*

$$\left\|\nabla f\left(\mathbf{P}_i^{(k+1)}\right)-\nabla f\left(\mathbf{P}_i^{(k)}\right)\right\| \le F_i\left\|\mathbf{P}_i^{(k+1)}-\mathbf{P}_i^{(k)}\right\| \quad \forall i.$$

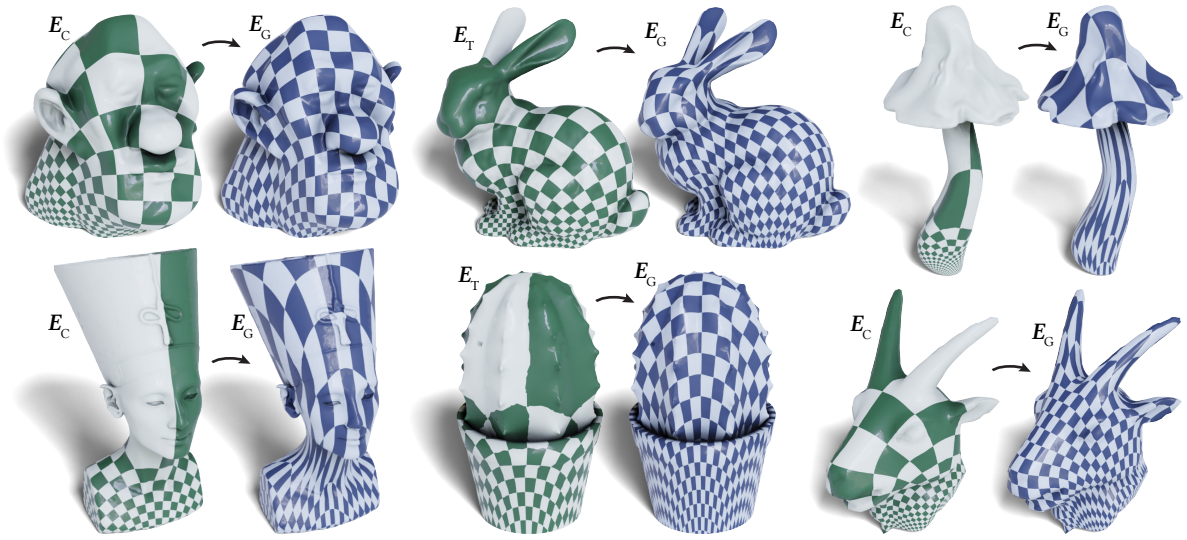


Figure 5. *Optimizing E_{G} with our splitting scheme to compute UV maps for a variety of surfaces. The surfaces are textured with a regular checkerboard texture. All generated UV maps are flip-free. The errors over the runtime of the optimizations are displayed in Figure 4.*

- *For $f = f_{\mathrm{G}}$, we have $F_i = (1 + \frac{\sqrt{d}}{C_i^{L^2}})$.*
- *For $f = f_{\mathrm{D}}$, we have $F_i = (1 + \frac{3\sqrt{d}}{C_i^{LG^4}})$.*

The constants are given by

- $C_i^L = -\frac{B_i}{2} + \frac{\sqrt{4+B_i^2}}{2}$ *and*
- C_i^{LG} *is the positive root of the quartic equation, i.e.,* $x^4 + B_i x^3 - 1 = 0$.

Moreover, $C_i^L \leq C_i^{LG} \leq 1$.

Based on Lemma 5.3 establishing an explicit local Lipschitz constant, we can now derive a few basic properties of our ADMM algorithm.

Proposition 5.4 (sufficient decrease property). *Suppose that*

$$\mu_i > \frac{1}{2}(-(w_i - 2\epsilon) + \sqrt{(w_i - 2\epsilon)^2 + 16\gamma w_i^2 F_i^2}),$$
$$h_i \geq \frac{4\gamma w_i^2 B_i^2}{\mu_i} + 2\epsilon,$$

and $0 < \epsilon < \frac{\min_i w_i}{2}$. *Let* $\{(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})\}_{k=0}^{\infty}$ *be the sequence of iterates generated by our ADMM algorithm, and denote* $\Phi(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})$ *by* Φ^k. *Then*

$$\Phi^{k+1} - \Phi^k \leq -\frac{1}{2}\lambda_{min}(L)\left\|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\right\|^2 - \sum_{t=1}^{m} \epsilon\left(\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2\right).$$

We are now ready to prove a global convergence result for our ADMM algorithm by characterizing the cluster point of the generated sequence.

Theorem 5.5 (global convergence of our splitting method). *If the set of KKT solutions for* (4.2) *that satisfy* (5.1) *is nonempty, then the augmented Lagrangian function* $\Phi(\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda})$ *is a Kurdyka–Łojasiewicz function, and hence the sequence generated by Algorithm* 4.1, $\{(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})\}_{k=0}^{\infty}$, *converges to a KKT point of* (4.2).

For a discussion on Kurdyka–Łojasiewicz functions, we refer the reader to [12] for details. The proofs for the statements in this section are given in Appendix B.

6. Implementation.

Algorithm 4.1 is the basic algorithm underlying our method, and we can analyze its properties theoretically (see section 5). The method we implement in code is slightly different, taking practical considerations into account.

Termination condition. There is no termination condition provided in Algorithm 4.1. We support a variety of termination conditions, such as a target energy, or no flipped triangles present (see Figure 3). The general-purpose termination condition, which we use in all experiments unless indicated, is a modified version of the primal and dual augmented Lagrangian errors [16, section 3.3.1], adjusted for our setting:

$$\begin{aligned}
\left(e_i^{\text{prim}}\right)^2 &= \left\| \left(G\mathbf{W}^{(k)}\right)_i - \mathbf{U}_i^{(k)}\mathbf{P}_i^{(k)} \right\|^2, \\
\left(e^{\text{prim}}\right)^2 &= \sum_{i=1}^{m} \left(e_i^{\text{prim}}\right)^2, \\
\left(e_i^{\text{dual}}\right)^2 &= \mu_i^2 \left\| \left(G\mathbf{W}^{(k)}\right)_i - \left(G\mathbf{W}^{(k-1)}\right)_i \right\|^2 \\
\left(e^{\text{dual}}\right)^2 &= \sum_{i=1}^{m} \left(e_i^{\text{dual}}\right)^2 .
\end{aligned} \tag{6.1}$$

We terminate the method if both of these errors are below the thresholds:

$$\begin{aligned}
e^{\text{prim}} &< \varepsilon_{\text{abs}}\sqrt{dm} + \varepsilon_{\text{rel}} \max\left\{ \left\|G\mathbf{W}^{(k)}\right\|, \left\|\mathbf{P}^{(k)}\right\| \right\}, \\
e^{\text{dual}} &< \varepsilon_{\text{abs}}\sqrt{dm} + \varepsilon_{\text{rel}} \left\|G^{\top}\mathbf{\Lambda}^{(k)}\right\| .
\end{aligned} \tag{6.2}$$

For most examples throughout this article, we choose $\varepsilon_{\text{abs}} = 10^{-6}, \varepsilon_{\text{rel}} = 10^{-5}$. For deformation experiments, we use $\varepsilon_{\text{abs}} = 5 \cdot 10^{-10}, \varepsilon_{\text{rel}} = 5 \cdot 10^{-9}$. Additionally, we add to the termination condition that all elements in the iterate $\mathbf{W}^{(k)}$ have to be flip-free.

Rescaling μ_i and $\mathbf{\Lambda}$. To speed up the optimization, we dynamically adjust the penalty parameters μ_i [16, section 3.4.1]. The goal of the rescaling algorithm is to keep e^{prim} and e^{dual} from (6.1) roughly equal. Thus,

- if $e_i^{\text{prim}} > \rho e_i^{\text{dual}}$, we multiply μ_i by $\frac{\rho}{2}$ and divide $\mathbf{\Lambda}_i^{(k)}$ by $\frac{\rho}{2}$;
- if $e_i^{\text{dual}} > \rho e_i^{\text{prim}}$, we divide μ_i by $\frac{\rho}{2}$ and multiply $\mathbf{\Lambda}_i^{(k)}$ by $\frac{\rho}{2}$,

where, in our implementation, we set $\rho = 5$. We employ a lower bound for μ_i of $\frac{1}{2}\mu_{\min}$, where $\mu_{\min}$ is the bound from section 5 computed with $B_i = \sqrt{5(1 + \|\nabla f(\mathbf{P}_i^{(k)})\|^2)}$, and F_i^2 is capped at $\varepsilon_m^{-1/4}$ (ε_m is the floating point machine epsilon). The changing μ_i and B_i as well as the lower bound of $\frac{1}{2}\mu_{\min}$ do not reflect the conditions of our convergence proof, but

such heuristic modifications to ADMM algorithms for actual implementations are used in practice; see, e.g., [16, section 3.4.1] for rescaling, and [109, section 6.1] for an example of gradient descent step sizes larger than suggested by the Lipschitz continuity bound (which is also where our $\mu_{\min}$ originates).

Since changing the penalties μ_i requires decomposition of L, we rescale sparingly: five times directly after initialization, and every $5(\frac{3}{2})^p$ iterations, where p is the number of past rescaling events. This way we rescale more in the beginning of the optimization, when the iterates are changing a lot, and less afterwards, when the iterates are not changing as much anymore. The initial penalty parameters (before rescaling) are set to $\mu_i = w_i$. h_i is set to its value from section 5, and disabled if we expect many flipped triangles in the input. γ is always set to 1 and ϵ to 0.

Initializing $\mathbf{U}, \mathbf{P}, \mathbf{\Lambda}$. The user does not need to supply $\mathbf{U}^{(0)}, \mathbf{P}^{(0)}, \mathbf{\Lambda}^{(0)}$. We can use the supplied $\mathbf{W}^{(0)}$ to initialize $\mathbf{U}, \mathbf{P}$ by employing a polar decomposition,

$$\left(G\mathbf{W}^{(0)}\right)_i = \mathbf{U}_i^{(0)}\mathbf{P}_i^{(0)} \,. \tag{6.3}$$

This is, in practice, computed using the singular value decomposition. For $(G\mathbf{W}^{(0)})_i = R_1 \Sigma R_2^\top$, we set

$$\begin{aligned} \mathbf{U}_i^{(0)} &= R_1 R_2^\top, \\ \mathbf{P}_i^{(0)} &= R_2 \Sigma R_2^\top \,. \end{aligned} \tag{6.4}$$

If $(G\mathbf{W}^{(0)})_i$ is flipped, then $R_1 R_2^\top$ is not a rotation matrix. In this case we multiply its last column by -1 such that $\det \mathbf{U}_i^{(0)} = 1$, and set $\Sigma = \varepsilon I$. To avoid numerical problems, we also set all values on the diagonal of Σ to ε should they be smaller. We set $\varepsilon = \varepsilon_m^{1/4}$ when optimizing E_{G}, and $\varepsilon = \varepsilon_m^{1/8}$ when optimizing E_{D}, where ε_m is the machine epsilon. $\Lambda^{(0)}$ is set to 0.

Our complete practical method is described in pseudocode form in Algorithm 6.1. We use IEEE double precision as our floating point type. The actual `C++` implementation built on libigl [49] will be publicly released under an open-source license after publication. We run our implementation on a 2.4 GHz Quad-Core Intel i5 MacBook Pro with 16 GB RAM.

The user needs to initialize our method with a map to a target mesh $\mathbf{W}^0$. This map does not need to be flip-free, however, it cannot be arbitrary (see section 9). We initialize, depending on the application, either with minimizers of E_{T} or E_{C}, which can be optimized very cheaply. Figure 4 shows the primal and dual errors over the entire optimization, as well as proxies for Conditions 5.1 and 5.2, for applications of Algorithm 6.1 to UV parametrization.

7. The symmetric gradient energy.

As a brief aside in the larger story of our optimization algorithm, we propose an alternative to the distortion energies mentioned in Appendix A, which we call the *symmetric gradient energy*. Although our main method applies to a broad class of distortion energies, we find that this alternative energy yields maps with favorable properties.

While f_{D} is symmetric with respect to inversion of its input matrix X, its gradient (A.2) is not. Moreover, the singularity in the gradient is rather strong ($\sim 1/x^3$).

Algorithm 6.1. Our implementation of Algorithm 4.1

method SplittingOptimization $\left(\mathbf{W}^{(0)}\right)$:
$\mathbf{U}^{(0)}, \mathbf{P}^{(0)} \leftarrow$ polar_decomposition$(\mathbf{W}^{(0)})$
$\mathbf{\Lambda}^{(0)} \leftarrow 0$
$\mu_i \leftarrow \max(\mu_{\min,i}, w_i) \quad \forall i$
decompose(L)
for $k \leftarrow 1, \ldots,$ max_iter **do**
  $\mathbf{W}^{(k)} \leftarrow \operatorname{argmin}_{\mathbf{W}, A\mathbf{W}=b} \Phi\left(\mathbf{W}, \mathbf{U}^{(k-1)}, \mathbf{P}^{(k-1)}, \mathbf{\Lambda}^{(k-1)}\right)$
  $\mathbf{U}^{(k)} \leftarrow \operatorname{argmin}_{\mathbf{U}} \Phi\left(\mathbf{W}^{(k)}, \mathbf{U}, \mathbf{P}^{(k-1)}, \mathbf{\Lambda}^{(k-1)}\right) + p\left(\mathbf{U}, \mathbf{U}^{(k-1)}\right)$
  $\mathbf{P}^{(k)} \leftarrow \operatorname{argmin}_{\mathbf{P}} \Phi\left(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}, \mathbf{\Lambda}^{(k-1)}\right)$
  $\mathbf{\Lambda}_i^{(k)} \leftarrow \mathbf{\Lambda}_i^{(k-1)} + (G\mathbf{W}^{(k)})_i - \mathbf{U}_i^{(k)}\mathbf{P}_i^{(k)} \quad \forall i$
  if rescale_at_iter(k) **then**
    $\mu_i \leftarrow$ rescale$(\mu_i, \mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)}) \quad \forall i$
    decompose(L)
  if termination_condition$(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})$ **then**
    break

Definition 7.1 (symmetric gradient energy). *The defining function* $f_{\mathrm{G}} : \mathbb{R}^{d\times d} \to \mathbb{R}$ *of the symmetric gradient energy is given by*

$$f_{\mathrm{G}}(X) \coloneqq \frac{1}{2}\|X\|^2 - \log\det X \,. \tag{7.1}$$

The symmetric gradient energy E_{G} *is defined as the flip-free generic distortion energy (3.2) with* $f = f_{\mathrm{G}}$.

Just like E_{D}, E_{G} is continuous and bounded for all flip-free maps. E_{G} is singular exactly when E_{D} is, just with a weaker singularity of $\sim \log x$ instead of $\sim 1/x$ (see also Appendix A.1). E_{G} is also invariant to rotations, as for any rotation matrix U, $f_{\mathrm{G}}(UX) = f_{\mathrm{G}}(X)$.

To our knowledge, E_{G} has not appeared in previous work on distortion energies in this form. E_{G} is, however, similar to other alternatives, chiefly among them the norm of the Hencky strain tensor [40] $\|\log X^T X\|^2$, which also features a logarithmic term. Other previous works discuss a strain energy that consists of only the logarithmic term of f_{G} [71]. E_{G} is, of course, also similar to E_{D}, which replaces the logarithmic term of E_{G} with an inverse term.

The gradient of E_{G}'s defining function is given by

$$\nabla f_{\mathrm{G}}(X) = X - X^{-\top} \,. \tag{7.2}$$

As X appears both as itself and as its inverse in the gradient of f_{G}, we call this energy the symmetric gradient energy to parallel the symmetric Dirichlet energy, where both appear in the function itself. The singularity in the gradient ($\sim 1/x$) is weaker than E_{D}'s ($\sim 1/x^3$).

E_{G} can be an alternative to other distortion energies in certain applications, depending on specific goals. Minimizers of E_{G} look more visually similar to the popular non-flip-free minimizers of E_{A} than minimizers of E_{D}. If the goal is a maximally rigid map that is still

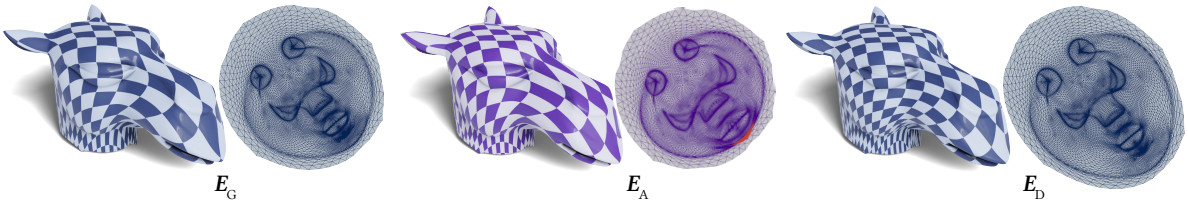


Figure 6. E_G (left) *exhibits more qualitative similarity to* E_A (center) *than* E_D (right). *The* L^2 *area distortions are, from left to right,* 0.000120, 0.000103, 0.000213 *(flipped triangles in red).*

flip-free, E_G is a valid choice. Additionally, minimizers of E_G can exhibit lower area distortion than minimizers of E_D, yielding a more faithful map in terms of area (see Figure 6). Because of these properties, we prefer using E_G for UV parametrization and volume correspondence (where the lower area distortion is a key feature), and E_D for deformation (where the weaker singularity of E_G can lead to distorted elements near the constrained parts of the mesh).

8. Results.

We use our splitting scheme to optimize distortion energies for three different applications: UV mapping, shape deformation, and volume correspondence.

8.1. UV maps.

A UV map of a triangle mesh $\mathbf{V} \subseteq \mathbb{R}^3$ is a map from $\mathbf{V}$ into $\mathbb{R}^2$, the UV space. UV maps have a variety of applications, such as texturing surfaces [5, section 6.4], quad meshing [13], machine learning on meshes [62], and more [78, 89]. We can compute a UV map by minimizing the distortion of a map from $\mathbf{V}$ into UV space. For applications such as texture mapping, it is especially important to have a low-distortion UV map, since high distortion will require higher-resolution images for texturing. Figures 1, 5 show our splitting method used to minimize E_G to arrive at a UV map. The surfaces are textured using a regular checkerboard texture to visualize the distortion of the UV map; the rendered checkerboard scale is manually set to attain qualitatively similar triangle sizes. In Figure 2 and Table 2 we compare our method with multiple previous works when optimizing E_D. We can see that for some examples the meshes are so challenging (as they contain a lot of branches and appendages that get squished down into a small area) that the previous methods were unable to produce a flip-free distortion minimizing optimization—an area where our method's robustness to flipped triangles enables us to compute results despite the difficulty. Figure 11 shows our method applied to a large parametrization data set.

8.2. Deformation.

Distortion energies can be used for deformation by constraining a part of the target mesh $\mathbf{W}$. We can constrain isolated vertices, as well as entire regions of the mesh, and then run our optimization method, initialized with the identity map. The few initially distorted (and potentially flipped) elements do not prevent our method from finding a solution.

In Figures 1, 7, 8 we deform a variety of surfaces and volumes by constraining vertices of the target mesh, and compare the results of the deformation using our method using E_D with the results of minimizing the ARAP energy E_A. Our method succeeds in producing natural-looking deformations while avoiding inverted elements. Figure 9 shows our method applied to interactive deformation: we created a user interface that allows fixing vertices of the target mesh and dragging them to the desired location. The displayed mesh is updated interactively during each iteration of our method. This results in a smooth interactive experience, as each iteration of our method is cheap to compute.

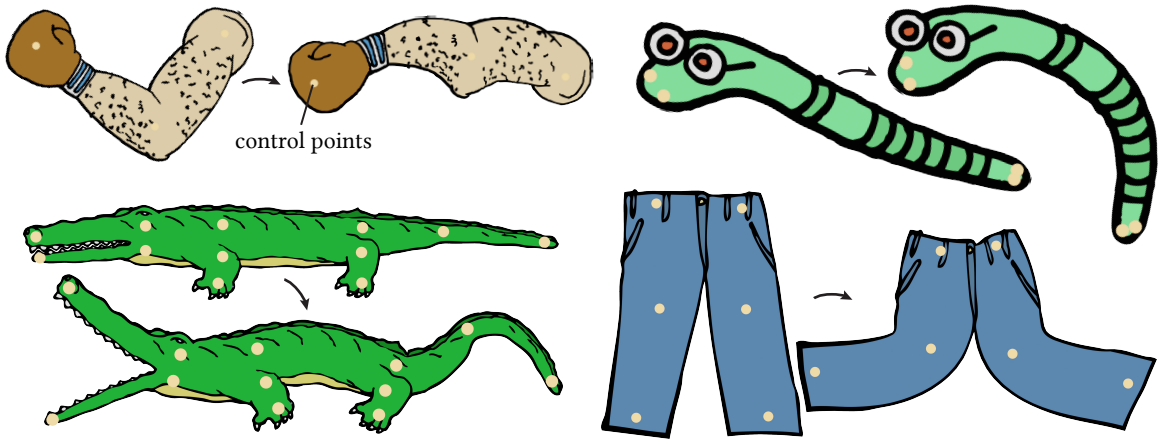


Figure 7. *Computing low-distortion deformation of surfaces in $\mathbb{R}^2$ by minimizing E_{D}, initializing with the identity map. The highlighted control points in the target mesh are fixed to the desired position, and our method is employed to minimize distortion.*

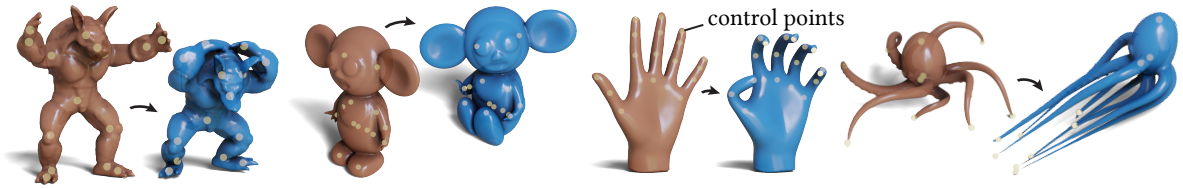


Figure 8. *Computing low-distortion deformation of volumes in $\mathbb{R}^3$ by minimizing E_{D}, initializing with the identity map. The highlighted control points in the target mesh are fixed to the desired position, and our method is employed to minimize distortion.*

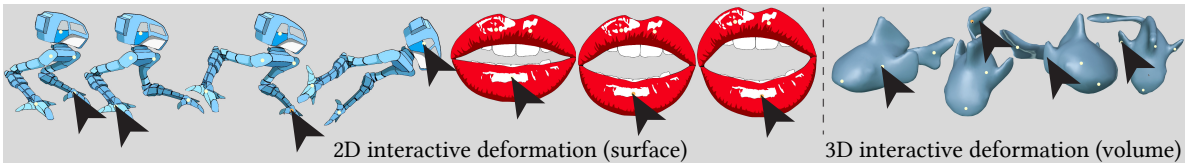


Figure 9. *Since each step of our iteration method can be evaluated cheaply, the method is well suited for interactive deformation. In our tool, the user picks which vertices they would like to constrain, and then drags them around while the shape deforms. For a video of this interactive application, see the accompanying supplemental files* M143305_02.mp4 [local/web 15.4MB], M143305_03.mp4 [local/web 9.3MB], and M143305_04.mp4 [local/web 11.5MB].

8.3. Volume correspondence. Our method can be used to compute a correspondence between the interior volumes of two given surfaces. To do this, we minimize the distortion of a map between the interior volume of the first surface (which has been tet-meshed [44, 96]), and the same volume with its boundary fixed to the second surface. The optimization is initialized with a minimizer of E_{T}, which can contain flipped tetrahedra; however, our method is able to arrive at a flip-free distortion-minimizing volume correspondence map nevertheless.

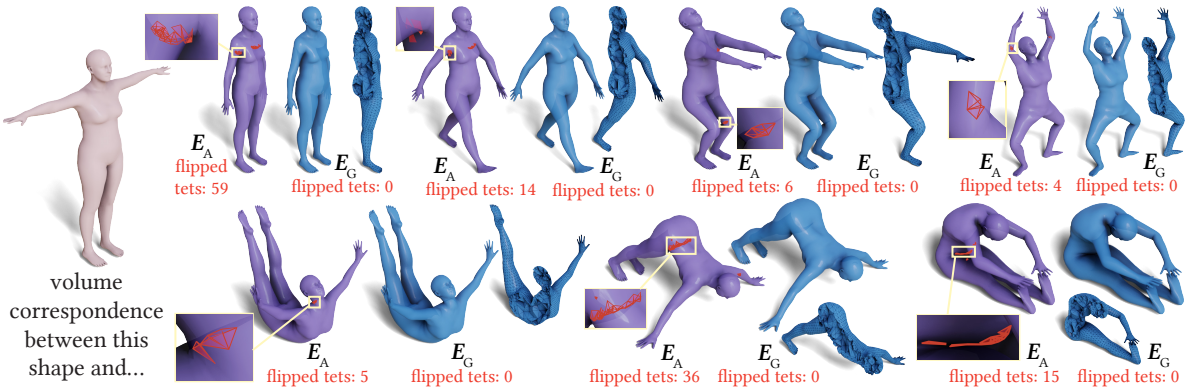


Figure 10. *Computing a correspondence between the tetrahedral mesh of a human* (far left)*, and the interior of a variety of surfaces whose boundaries correspond to the original human. Correspondences computed with our method using E_G exhibit no flipped tetrahedra* (top)*, while correspondences computed using E_A can contain flipped tetrahedra* (bottom, highlighted in red).

Figure 10 shows our method applied to compute correspondences between the interior of a human and a variety of other surfaces that show the same human, but in a different position. Figure 1 shows a volume correspondence between two different configurations of a teddy bear. In both cases, our method produces flip-free distortion-minimizing volume correspondences. Surface correspondences for applying our method can be computed, e.g., using [29].

9. Limitations. There are a few scenarios in which our method can fail. We cannot initialize our method with arbitrary input (see Figure 12, left), which can cause $\|\nabla f(\mathbf{P}_i^{(k)})\|$ to grow very large. In this case, our method will not converge to a flip-free optimum; this can prevent us from initializing with E_C in the presence of too many flips (this is the case, e.g., for the tree example in Table 2). In Figure 11, a few inputs fail to yield a flip-free parametrization.

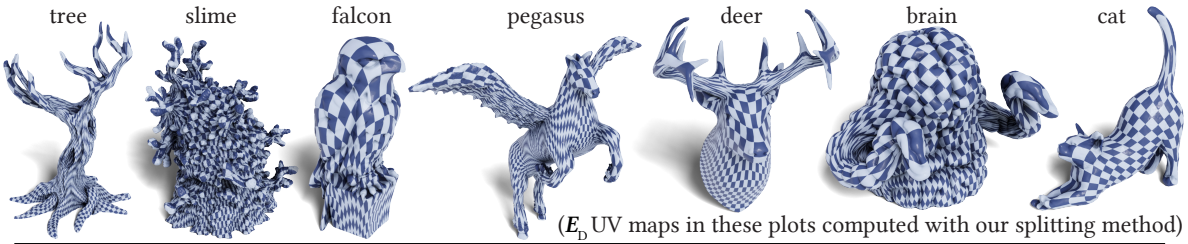
While minimizers of E_D are guaranteed to be flip-free, this does not mean that the maps will be bijective. As with previous methods that employ E_D, one often obtains bijective maps in practice, although this is not guaranteed. A k-cover of a triangle mesh can be completely flip-free, while also failing to be locally injective (see Figure 12, center). This applies to all methods which optimize flip-free energies that are not specifically bijective, such as E_D. Related works propose solutions to this problem [34, 98].

Our method can fail when constraints on $\mathbf{W}$ are imposed that constitute a very large deformation from the initial target mesh $\mathbf{W}^{(0)}$. An example of this can be seen in Figure 12, right. The deformation application has the additional limitation that the termination tolerances need to be set lower than for other applications to achieve good results (although this can be somewhat remedied by initializing with minimizers of E_A.

Theorem 5.5 guarantees convergence, assuming exact arithmetic. The implementation on the computer uses floating point arithmetic, which is not exact, which can result in elements that are flipped for numerical reasons. If such numerical issues occur while the optimization is still making progress, the method might recover, like it does for maps with flipped elements such as in Figure 3. If this happens when the method can no longer make useful progress, the method will fail to terminate (this is a known issue with parametrization methods [95]).

Table 2

Comparing the runtime of UV maps generated with our method to the previous methods of SLIM [84], AKVF [22], and PP [59]. Since each of these related works uses their own termination condition, they do not all arrive at a parametrization with the same E_D. To compare against each previous method as intended by its authors, we run the publicly available implementation of the method until it satisfies its own termination condition (or is aborted after 150 minutes). We then measure E_D of the previous method's UV map (after rescaling it to match the total area of the original mesh), and run our own algorithm to produce a flip-free map matching the previous method's E_D up to a tolerance of 10^{-6}. Thus the runtimes in this table are plotted in pairs: a previous method, as well as our algorithm set to produce a map with the same distortion. We initialize our method with minimizers of E_T and E_C, and report the best time. If the previous method did not terminate successfully, our method is run with default termination conditions (which, in general, are set to produce lower-energy results than previous methods' termination conditions). Values are rounded to three significant digits.



(E_D UV maps in these plots computed with our splitting method)

	m	$\frac{\text{min. area}}{\text{max. area}}$	SLIM	Ours (match SLIM E_D)	AKVF	Ours (match AKVF E_D)	PP	Ours (match PP E_D)
camel	3.58k	0.00191	0.246 s	0.134 s†	0.947 s	3.16 s†	0.115 s	3.20 s†
triceratops	5660	$3.27 \cdot 10^{-7}$	0.324 s	0.233 s†	0.968 s	2.25 s†	–‡	0.403 s†
cow	5.80k	0.00132	0.346 s	0.212 s†	1.53 s	1.05 s†	0.207 s	1.08 s†
tooth	9129	$3.27 \cdot 10^{-7}$	0.426 s	0.307 s*	1.36 s	0.367 s*	0.252 s	0.395 s*
hand	9.36k	0.00174	–	1.95 s†	1.31 s	1.51 s†	0.516 s	1.50 s†
deer	10.9k	0.000179	–	6.10 s*	–	6.76 s*	–‡	240 s†
horse	39.7k	0.000429	1.46 s	1.23 s†	7.64 s	10.8 s†	1.04 s	10.9 s†
bread	49.9k	0.000760	2.07 s	2.81 s†	14.4 s	2.84 s†	1.10 s	2.81 s†
falcon	51.5k	0.000180	–	7.71 s*	–	7.48 s*	1.22 s	9.05 s*
cat	90.0k	0.00377	3.54 s	19.4 s*	10.5 s	15.7 s*	3.62 s	25.4 s*
car	97.6k	$5.25 \cdot 10^{-6}$	3.83 s	11.0 s†	15.5 s	11.2 s†	1.99 s	11.1 s†
brain	152k	0.00107	–	308 s*	–	308 s*	15.7 s	16.8 s*
strawberry	313k	$1.81 \cdot 10^{-6}$	14.8 s	55.3 s*	45.1 s	59.8 s*	9.57 s	60.5 s*
slime	567k	0.000980	–	1140 s*	112 s	38.4 s*	53.7 s	42.9 s*
tree	630k	$5.84 \cdot 10^{-5}$	–	2080 s*	––	2090 s*	187 s	76.0 s*
pegasus	2390k	$1.82 \cdot 10^{-10}$	–	859 s*	–	866 s*	–	862 s*

– a previous work was unable to find a distortion-minimizing flip-free mapping (it errored, or flips were present in the result).
–– the algorithm terminated, but with a very high energy, which is counted as unsuccessful.
∗ our method initialized with a minimizer of E_T.
† our method initialized with a minimizer of E_C.
‡ previous method will sometimes work, and sometimes fail; thus reported as fail here.

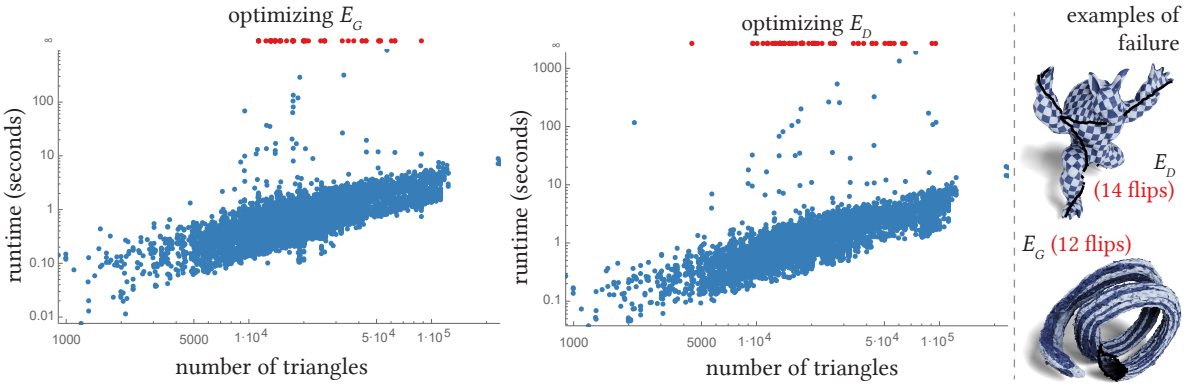


Figure 11. *Computing UV parametrizations for all meshes in the D1, D2, and D3 datasets [59] (modified to exclude degenerate meshes), consisting of 15525 meshes. The method was initialized with E_T, run with a maximal iteration count of 100000, and with termination tolerances $\varepsilon_{abs} = 5 \cdot 10^{-5}, \varepsilon_{rel} = 5 \cdot 10^{-4}$. Successful meshes are shown in blue, failed meshes (0.38% for E_G, 0.53% for E_D) in red.*

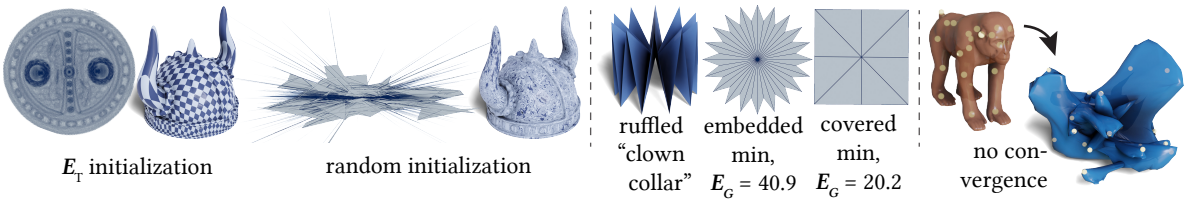


Figure 12. *If our method is initialized with a bad initial target mesh* $\mathbf{W}$*, it will not converge like it would with a good initial target mesh (e.g., the minimizer of E_T)* (left). *While our method will always produce a flip-free map, not all flip-free maps are bijective: the k-cover of this ruffled high-valence vertex has lower distortion than the bijective map* (center). *If the target mesh* $\mathbf{W}$ *is constrained to a deformation that is very far from the initial mesh, our method can fail to converge* (right).

10. Conclusion.

In this paper, we have proposed a new splitting scheme for the optimization of flip-free distortion energies, discussed the convergence behavior of the resulting ADMM algorithm under certain conditions, and demonstrated its utility in a variety of applications.

There are opportunities for future work in many directions. On the application side, our method could be used for other applications where flip-free distortion-minimizing mappings are required, such as in elasticity simulation combined with contact mechanics, for example, in the context of an efficient solver such as projective dynamics [15]; or to deform shapes with suitability for fabrication in mind [14]. On the implementation side, our method could be considerably sped up by implementing some of our parallelized instructions in the $\mathbf{P}$ and $\mathbf{U}$ optimization step in a way that exploits simultaneous execution capabilities of modern CPUs such as SSE or AVX, similar related approaches [64] for the optimization of E_A [49, `polar_svd3x3.h`]. On the algorithm side, further approaches to improve the performance of the ADMM can be employed. There is a lot of recent work on the topic, and some of it might be able to speed up our splitting method.

Appendix.

A. Distortion energies. This appendix discusses distortion energies appearing in this article that have been featured extensively in previous work.

A.1. The symmetric Dirichlet energy.

Definition A.1 (symmetric Dirichlet energy [98]). *The defining function* $f_{\mathrm{D}} : \mathbb{R}^{d\times d} \to \mathbb{R}$ *of the symmetric Dirichlet energy is*

$$f_{\mathrm{D}}(X) \coloneqq \frac{1}{2}\left(\|X\|^2 + \left\|X^{-1}\right\|^2\right) . \tag{A.1}$$

The symmetric Dirichlet energy E_{D} *is defined as the flip-free generic distortion energy* (3.2) *with* $f = f_{\mathrm{D}}$.

E_{D} is finite and continuous on all maps whose Jacobian has positive determinant. By the definition of (3.2), a negative determinant leads to infinite E_{D} (due to χ_+), and since f_{D} has a singularity for matrices with zero determinant, there is a barrier preventing determinants from approaching zero. Hence, minimizers of E_{D} are flip-free (unless overconstrained). Furthermore, E_{D} is invariant to rotations: for a rotation matrix U, $f_{\mathrm{D}}(UX) = f_{\mathrm{D}}(X)$.

The gradient of the defining function is given by

$$\nabla f_{\mathrm{D}}(X) = X - X^{-\top}X^{-1}X^{-\top} . \tag{A.2}$$

Unlike f_{D} itself, its gradient is not symmetric with respect to inverting its argument. The gradient also features a stronger singularity ($\sim 1/x^3$) than the defining function ($\sim 1/x$).

Our optimization method reproduces the output of previous methods [22, 59, 84] when optimizing E_{D}. Figure 2 shows our method as well as previous methods used to compute UV maps by minimizing E_{D}; the results visually match. Unlike these previous methods, we will be able to prove under which conditions our approach converges.

A.2. Noninjective distortion energies.

Beyond flip-free energies like the symmetric Dirichlet energy, many important energies allow elements to invert. Although their optima might not be desirable as final results, they have a large advantage over flip-free energies: they are usually much easier to optimize, thanks to linearity or a lack of singularities. Since our algorithm is resilient to initial iterates that contain flips, we can use optima of these simpler energies as initializers (with exceptions; see section 9).

A.2.1. Tutte's energy.

Definition A.2 (Tutte's energy [108]). *Tutte's energy for the target mesh* $\mathbf{W}$ *is given by*

$$E_{\mathrm{T}}(\mathbf{W}) \coloneqq \sum_{edges\ (i,j)} \frac{1}{\|\mathbf{V}_i - \mathbf{V}_j\|}\|\mathbf{W}_i - \mathbf{W}_j\|^2 .$$

While Tutte did not define his energy exactly as written above, that definition can be found in recent references [48].

Minimizers of E_{T} for triangulated surfaces are flip-free if all boundary vertices are constrained to a convex shape [108]. E_{T} is a quadratic energy which can be efficiently minimized

by solving a linear system without initialization. Hence, minimizing E_{T} is a popular strategy for generating initial flip-free parametrizations to launch additional line-search-based optimization steps that further reduce distortion [22, 59, 84, 98].

Minimizers of E_{T} for *tetrahedralized volumes*, however, are not in general flip-free, even if all boundary vertices are constrained to a convex shape. This is an obstacle for optimization methods that need to start with a flip-free map. Our method does not automatically fail if the initial map contains flipped elements, and can thus use E_{T} even for volumes.

A.2.2. Conformal energy. As a contrast to Tutte's energy, one can construct quadratic energies built on estimates of the derivative of a surface/volume map, sensitive to conformal (angle-based) geometry. One popular choice is the conformal energy.

Definition A.3 (Conformal energy [67]). *The conformal energy for the target mesh* $\mathbf{W}$ *is*

$$E_{\mathrm{C}}(\mathbf{W}) \coloneqq \frac{1}{2}\sum_{i=1}^{m} w_i \left\|(G\mathbf{W})_i\right\|^2 - A(\mathbf{W}) \ ,$$

where $A(\mathbf{W})$ *is the area of the target mesh* $\mathbf{W}$.

Similarly to E_{T}, E_{C} is quadratic in $\mathbf{W}$. Unlike E_{T}, however, minimizers of E_{C} are not guaranteed to be flip-free for surfaces.

Several papers propose ways to discretize and optimize E_{C} in practice (see section 2.1). In this work, we employ the method [86], which efficiently minimizes E_{C} with free boundary and minimal area distortion.

A.2.3. As-rigid-as-possible energy. The linear energies above do not directly measure the deviation of a map from being *rigid* The ARAP energy is specifically designed to be sensitive to nonrigidity.

Definition A.4 (ARAP Energy [60, 101]). *The ARAP energy's defining function is*

$$f_{\mathrm{A}}(X) \coloneqq \frac{1}{2}\left\|X - \operatorname{rot} X\right\|^2 \ ,$$

where $\operatorname{rot} X$ *isolates the rotational part of a matrix* X *by solving a Procrustes problem* [36],

$$\operatorname{rot}(X) \coloneqq \operatorname*{argmin}_{R\in \mathrm{SO}(d)} \left\|R - X\right\|^2 \ .$$

The ARAP energy E_{A} *is defined as the generic distortion energy with flips* (3.1) *and* $f = f_{\mathrm{A}}$.

In this article we employ the local-global solver with per-element discretization [60] as implemented by libigl [49], which we denote by E_{A}. We run the optimization until the relative error between two subsequent iterates is less than 10^{-6}, but not more than 150 minutes.

E_{A} is a popular distortion energy; it produces results that are reminiscent of elasticity, while being cheap to optimize. Its minimizers are, however, not always flip-free.

B. Additional calculations for the convergence proof. This appendix contains proofs for the convergence analysis in section 5.

Proof of Lemma 5.3. Let $\mathbf{P}_i = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^\top$ be the eigenvalue decomposition of $\mathbf{P}_i$. We consider E_{G} first ($f = f_{\mathrm{G}}$). Recall that $f(\mathbf{P}_i) = \frac{1}{2}\|\mathbf{P}_i\|^2 - \log\det\mathbf{P}_i$. Hence,

$$\|\nabla f(\mathbf{P}_i)\|^2 = \|\mathbf{P}_i - {\mathbf{P}_i}^{-1}\|^2 = \|\mathbf{\Sigma} - \mathbf{\Sigma}^{-1}\|^2 = \sum_{i=1}^{d}(\lambda_i - \lambda_i^{-1})^2,$$

where $\mathbf{\Sigma} = \mathrm{Diag}([\lambda_1, \ldots, \lambda_d])$ and $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_d$. By Condition 5.1, we have

$$\lambda_1(\mathbf{P}_i) \le C_i^L \triangleq -\frac{B_i}{2} + \frac{\sqrt{4 + B_i^2}}{2}. \tag{B.1}$$

Next, we aim to use the term $\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\|$ to bound $\|\nabla f(\mathbf{P}_i^{(k+1)}) - \nabla f(\mathbf{P}_i^{(k)})\|$, using the conditions $\lambda_1(\mathbf{P}_i^{(k+1)}) \le C_i^L$ and $\lambda_1(\mathbf{P}_i^{(k)}) \le C_i^L$, i.e., (B.1).

$$\begin{aligned}\left\|\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \nabla f\left(\mathbf{P}_i^{(k)}\right)\right\| &= \left\|\left(\mathbf{P}_i^{(k+1)} - {\mathbf{P}_i^{(k+1)}}^{-1}\right) - \left(\mathbf{P}_i^{(k)} - {\mathbf{P}_i^{(k)}}^{-1}\right)\right\| \\ &\le \left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\| + \left\|{\mathbf{P}_i^{(k+1)}}^{-1} - {\mathbf{P}_i^{(k)}}^{-1}\right\|.\end{aligned}$$

We proceed to bound $\|{\mathbf{P}_i^{(k+1)}}^{-1} - {\mathbf{P}_i^{(k)}}^{-1}\|$,

$$\begin{aligned}\left\|{\mathbf{P}_i^{(k+1)}}^{-1} - {\mathbf{P}_i^{(k)}}^{-1}\right\| &\le \sqrt{d}\left\|{\mathbf{P}_i^{(k+1)}}^{-1} - {\mathbf{P}_i^{(k)}}^{-1}\right\|_2 = \sqrt{d}\left\|{\mathbf{P}_i^{(k)}}^{-1}\left(\mathbf{P}_i^{(k)} - \mathbf{P}_i^{(k+1)}\right){\mathbf{P}_i^{(k+1)}}^{-1}\right\|_2 \\ &\le \sqrt{d}\left\|{\mathbf{P}_i^{(k)}}^{-1}\right\|_2\left\|\mathbf{P}_i^{(k)} - \mathbf{P}_i^{(k+1)}\right\|_2\left\|{\mathbf{P}_i^{(k+1)}}^{-1}\right\|_2 \le \frac{\sqrt{d}}{{C_i^L}^2}\left\|\mathbf{P}_i^{(k)} - \mathbf{P}_i^{(k+1)}\right\|_2,\end{aligned}$$

where the last inequality follows from the spectral norm of the inverse coming from the first eigenvalue.

Combining the above two inequalities, we find that

$$\left\|\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \nabla f\left(\mathbf{P}_i^{(k)}\right)\right\| \le \left(1 + \frac{\sqrt{d}}{{C_i^L}^2}\right)\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|.$$

Following a similar argument, we can also derive the explicit local Lipschitz constant for E_{D} ($f = f_{\mathrm{D}}$). Recall that $f(\mathbf{P}_i) = \frac{1}{2}\left(\|\mathbf{P}_i\|^2 + \|\mathbf{P}_i^{-1}\|^2\right)$. Hence,

$$\|\nabla f(\mathbf{P}_i)\|^2 = \|\mathbf{P}_i - {\mathbf{P}_i}^{-3}\|^2 = \|\mathbf{\Sigma} - \mathbf{\Sigma}^{-3}\|^2 = \sum_{i=1}^{d}(\lambda_i - \lambda_i^{-3})^2.$$

By Condition 5.1 we have $\lambda_1(\mathbf{P}_i) \le C_i^{LG}$, i.e., (B.1), where the C_i^{LG} are the positive roots of the quartic equation $x^4 + B_i x^3 - 1 = 0$. Moreover, $C_i^L \le C_i^{LG} \le 1$ (see Figure 13).

$$\begin{aligned}\left\|\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \nabla f\left(\mathbf{P}_i^{(k)}\right)\right\| &= \left\|\left(\mathbf{P}_i^{(k+1)} - {\mathbf{P}_i^{(k+1)}}^{-3}\right) - \left(\mathbf{P}_i^{(k)} - {\mathbf{P}_i^{(k)}}^{-3}\right)\right\| \\ &\le \left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\| + \left\|{\mathbf{P}_i^{(k+1)}}^{-3} - {\mathbf{P}_i^{(k)}}^{-3}\right\|.\end{aligned}$$

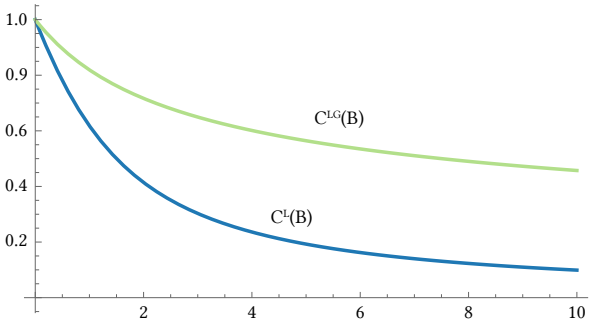


Figure 13. *A plot of the bounds C_i^L and C_i^{LG} with respect to the maximal gradient norm B_i. One can see that $C_i^L \leq C_i^{LG} \leq 1$.*

Similarly to our proof for the symmetric gradient energy, we next bound $\|\mathbf{P}_i^{(k+1)^{-3}} - \mathbf{P}_i^{(k)^{-3}}\|$:

$$\begin{aligned}\left\|\mathbf{P}_i^{(k+1)^{-3}} - \mathbf{P}_i^{(k)^{-3}}\right\| &\leq \sqrt{d}\left\|\mathbf{P}_i^{(k+1)^{-3}} - \mathbf{P}_i^{(k)^{-3}}\right\|_2 \\ &\leq \sqrt{d}\left\|\mathbf{P}_i^{(k+1)^{-3}}\left(\mathbf{P}_i^{(k+1)^3} - \mathbf{P}_i^{(k)^3}\right)\mathbf{P}_i^{(k)^{-3}}\right\|_2.\end{aligned}$$

We observe that

$$\begin{aligned}&\mathbf{P}_i^{(k+1)^3} - \mathbf{P}_i^{(k)^3} \\ &\quad= \mathbf{P}_i^{(k+1)^2}\left(\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right) + \mathbf{P}_i^{(k+1)}\left(\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right)\mathbf{P}_i^{(k)} + \left(\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right)\mathbf{P}_i^{(k)^2}.\end{aligned}$$

Based on the above equality, we find that

$$\begin{aligned}&\left\|\mathbf{P}_i^{(k+1)^{-3}}\left(\mathbf{P}_i^{(k+1)^3} - \mathbf{P}_i^{(k)^3}\right)\mathbf{P}_i^{(k)^{-3}}\right\|_2 \\ &\quad\leq \left\|\mathbf{P}_i^{(k+1)^{-1}}\left(\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right)\mathbf{P}_i^{(k)^{-3}}\right\|_2 + \left\|\mathbf{P}_i^{(k+1)^{-2}}\left(\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right)\mathbf{P}_i^{(k)^{-2}}\right\|_2 \\ &\qquad+ \left\|\mathbf{P}_i^{(k+1)^{-3}}\left(\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right)\mathbf{P}_i^{(k)^{-1}}\right\|_2 \\ &\quad\leq \frac{3}{C_i^{LG^4}}\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|.\end{aligned}$$

Combining all the statements derived above, we conclude

$$\left\|\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \nabla f\left(\mathbf{P}_i^{(k)}\right)\right\| \leq \left(1 + \frac{3\sqrt{d}}{C_i^{LG^4}}\right)\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|,$$

which proves the lemma. ∎

Proof of Proposition 5.4. We begin by deriving a sufficient decrease property for the augmented Lagrangian function. The core strategy here is to use the primal blocks $(\mathbf{W}, \mathbf{U}, \mathbf{P})$ to bound the dual variable $\mathbf{\Lambda}$.

$$\begin{aligned}\Phi^{k+1} - \Phi^k = &\underbrace{\Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k+1)}) - \Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k)})}_{(a)} \\ &+ \underbrace{\Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k)}) - \Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})}_{(b)} \\ &+ \underbrace{\Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)}) - \Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})}_{(c)} \\ &+ \underbrace{\Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)}) - \Phi(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})}_{(d)}.\end{aligned}$$

Focusing on the dual update,

$$\begin{aligned}(a) &= \sum_{i=1}^{m} \frac{\mu_i}{2} \left(\left\| \left(G\mathbf{W}^{(k+1)} \right)_i - \mathbf{U}_i^{(k+1)} \mathbf{P}_i^{(k+1)} + \mathbf{\Lambda}_i^{(k+1)} \right\|^2 - \left\| \mathbf{\Lambda}_i^{(k+1)} \right\|^2 \right) \\ &\quad - \sum_{i=1}^{m} \frac{\mu_i}{2} \left(\left\| \left(G\mathbf{W}^{(k+1)} \right)_i - \mathbf{U}_i^{(k+1)} \mathbf{P}_i^{(k+1)} + \mathbf{\Lambda}_i^{(k)} \right\|^2 - \left\| \mathbf{\Lambda}_i^{(k)} \right\|^2 \right) \\ &= \sum_{i=1}^{m} \frac{\mu_i}{2} \left(\left\| \mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)} + \mathbf{\Lambda}_i^{(k+1)} \right\|^2 - \left\| \mathbf{\Lambda}_i^{(k+1)} \right\|^2 \right) \\ &\quad - \sum_{i=1}^{m} \frac{\mu_i}{2} \left(\left\| \mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)} + \mathbf{\Lambda}_i^{(k)} \right\|^2 - \left\| \mathbf{\Lambda}_i^{(k)} \right\|^2 \right) \\ &= \sum_{i=1}^{m} \mu_i \left\| \mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)} \right\|^2.\end{aligned}$$

To use the primal blocks to bound the dual update (a), we first write down the optimality condition with respect to $\mathbf{P}^{(k+1)}$:

$$\begin{aligned}\mathbf{P}^{(k+1)} &= \operatorname*{argmin}_{\mathbf{P} \in (\mathcal{S}_+^d)^m} \Phi\left(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}, \mathbf{\Lambda}^{(k)} \right) \\ \Rightarrow\ 0 &= w_i \nabla f\left(\mathbf{P}_i^{(k+1)} \right) + \mu_i \operatorname{symm}\left({\mathbf{U}_i^{(k+1)}}^\top \left(\mathbf{U}_i^{(k+1)} \mathbf{P}_i^{(k+1)} - \left(G\mathbf{W}^{(k+1)} \right)_i - \mathbf{\Lambda}_i^{(k)} \right) \right) \quad \forall i \\ \Rightarrow 0 &= w_i \nabla f\left(\mathbf{P}_i^{(k+1)} \right) - \mu_i \operatorname{symm}\left({\mathbf{U}_i^{(k+1)}}^\top \mathbf{\Lambda}_i^{(k+1)} \right) \quad \forall i.\end{aligned}$$

Then, we have

$$\frac{w_i}{\mu_i} \mathbf{U}_i^{(k+1)} \nabla f\left(\mathbf{P}_i^{(k+1)} \right) = \frac{1}{2} \left(\mathbf{\Lambda}_i^{(k+1)} + \mathbf{U}_i^{(k+1)} {\mathbf{\Lambda}_i^{(k+1)}}^\top \mathbf{U}_i^{(k+1)} \right) \quad \forall i.$$

Thus, we can use $(\mathbf{U}, \mathbf{P})$ to bound (a). By Condition 5.2,

$$\left\|\mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)}\right\|^2 \leq \gamma \left\|\frac{1}{2}\left(\mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)}\right) + \frac{1}{2}\left(\mathbf{U}_i^{(k+1)}\mathbf{\Lambda}_i^{(k+1)\top}\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\mathbf{\Lambda}_i^{(k)\top}\mathbf{U}_i^{(k)}\right)\right\|^2.$$

Thus,

$$\begin{aligned}
&\left\|\mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)}\right\|^2 \\
&\quad\leq \frac{\gamma w_i^2}{\mu_i^2}\left\|\mathbf{U}_i^{(k+1)}\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \mathbf{U}_i^{(k)}\nabla f\left(\mathbf{P}_i^{(k)}\right)\right\|^2 \\
&\quad= \frac{\gamma w_i^2}{\mu_i^2}\left\|\mathbf{U}_i^{(k+1)}\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \mathbf{U}_i^{(k)}\nabla f\left(\mathbf{P}_i^{(k+1)}\right) + \mathbf{U}_i^{(k)}\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \mathbf{U}_i^{(k)}\nabla f\left(\mathbf{P}_i^{(k)}\right)\right\|^2 \\
&\quad\leq \frac{2\gamma w_i^2}{\mu_i^2}\left(B_i^2\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \left\|\nabla f\left(\mathbf{P}_i^{(k+1)}\right) - \nabla f\left(\mathbf{P}_i^{(k)}\right)\right\|^2\right) \\
&\quad\leq \frac{2\gamma w_i^2 B_i^2}{\mu_i^2}\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \frac{2\gamma w_i^2 F_i^2}{\mu_i^2}\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2,
\end{aligned}$$

where the last inequality follows from lemma 5.3. Based on the above analysis,

$$\begin{aligned}
(a) &= \sum_{i=1}^m \mu_i \left\|\mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)}\right\|^2 \\
&\leq \sum_{i=1}^m \frac{2\gamma w_i^2 B_i^2}{\mu_i}\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \frac{2\gamma w_i^2 F_i^2}{\mu_i}\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2.
\end{aligned}$$

We continue by bounding the terms (b), (c), (d). As $\Phi(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}, \mathbf{\Lambda}^{(k)})$ is $(w_i+\mu_i)$-strongly convex for $\mathbf{P}_i$, we have,

$$(b) \leq -\sum_{i=1}^m \frac{w_i + \mu_i}{2}\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2.$$

See [72, Theorem 2.1.8] for details. Since $\mathbf{U}^{(k+1)}$ is the global optimal solution of $\Phi(\mathbf{W}^{(k+1)}, \mathbf{U}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)}) + \sum_{i=1}^m \frac{h_i}{2}\|\mathbf{U}_i - \mathbf{U}_i^{(k)}\|^2$, we obtain

$$(c) \leq -\sum_{i=1}^m \frac{h_i}{2}\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2.$$

Similarly, we can derive the associated sufficient decrease term for $\mathbf{W}$, i.e.,

$$(d) \leq -\frac{1}{2}\lambda_{\min}(L)\left\|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\right\|^2.$$

By summing up all the inequalities for $(a), (b), (c), (d)$,

$$\begin{aligned}\Phi^{k+1} - \Phi^k \leq & -\sum_{i=1}^{m}\left(\frac{h_i}{2} - \frac{2\gamma w_i^2 B_i^2}{\mu_i}\right)\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 \\ & -\sum_{i=1}^{m}\left(\frac{w_i+\mu_i}{2} - \frac{2\gamma w_i^2 F_i^2}{\mu_i}\right)\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2 \\ & -\frac{1}{2}\lambda_{\min}(L)\left\|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\right\|^2.\end{aligned}$$

We now apply the conditions $\mu_i > -\frac{1}{2}(w_i - 2\epsilon) + \frac{1}{2}\sqrt{(w_i-2\epsilon)^2 + 16\gamma w_i^2 F_i^2}$ and $h_i \geq \frac{4\gamma w_i^2 B_i^2}{\mu_i} + 2\epsilon$ to arrive at

$$\begin{aligned}\Phi^{k+1} - \Phi^k \leq & -\frac{1}{2}\lambda_{\min}(L)\left\|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\right\|^2 \\ & -\sum_{t=1}^{m}\epsilon\left(\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2\right),\end{aligned} \tag{B.2}$$

which proves the statement of the theorem. ∎

Proof of Theorem 5.5. There are four core steps to complete this proof.

- *Step* 1*:* Show that the sequence $\{(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})\}_{k=0}^{\infty}$ is bounded.

The boundedness of the sequence $\{\mathbf{P}^{(k)}\}_{k=0}^{\infty}$ follows directly from Condition 5.1. $\mathbf{U}_i$ is a rotation matrix and thus bounded. Recall that

$$\left\|\mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)}\right\|^2 \leq \frac{2\gamma w_i^2 B_i^2}{\mu_i^2}\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \frac{2\gamma w_i^2 F_i^2}{\mu_i^2}\left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2. \tag{B.3}$$

Therefore, we can conclude that the dual variable $\mathbf{\Lambda}$ is bounded. Using the update rule for $\mathbf{W}$ directly gives a bound for $\mathbf{W}$. Hence, the sequence $\{(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})\}_{k=0}^{\infty}$ is bounded, and thus a cluster point exists.

- *Step* 2*:* Prove that $\lim_{k\to+\infty} \|\mathbf{U}^{(k+1)} - \mathbf{U}^{(k)}\|^2 + \|\mathbf{P}^{(k+1)} - \mathbf{P}^{(k)}\|^2 + \|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\|^2 + \|\mathbf{\Lambda}^{(k+1)} - \mathbf{\Lambda}^{(k)}\|^2 = 0$, where the squared norm of $\mathbf{A}$ indicates the appropriate sum over all the squared norms of the $\mathbf{A}_i$.

Suppose that $(\mathbf{W}^*, \mathbf{U}^*, \mathbf{P}^*, \mathbf{\Lambda}^*)$ is a cluster point of the sequence $\{(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})\}_{k=0}^{\infty}$. Let $\{(\mathbf{W}^{(k_i)}, \mathbf{U}^{(k_i)}, \mathbf{P}^{(k_i)}, \mathbf{\Lambda}^{(k_i)})\}$ be a convergent subsequence such that

$$\lim_{i\to+\infty}\left(\mathbf{W}^{(k_i)}, \mathbf{U}^{(k_i)}, \mathbf{P}^{(k_i)}, \mathbf{\Lambda}^{(k_i)}\right) = (\mathbf{W}^*, \mathbf{U}^*, \mathbf{P}^*, \mathbf{\Lambda}^*).$$

By summing (B.2) from $k = 0$ to $k = k_i - 1$, we have

$$\begin{aligned}&\Phi(\mathbf{W}^{(k_i)}, \mathbf{U}^{(k_i)}, \mathbf{P}^{(k_i)}, \mathbf{\Lambda}^{(k_i)}) - \Phi(\mathbf{W}^{(0)}, \mathbf{U}^{(0)}, \mathbf{P}^{(0)}, \mathbf{\Lambda}^{(0)}) \\ &\leq -\frac{1}{2}\lambda_{\min}(L)\sum_{k=0}^{k_i-1}\left\|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\right\|^2 - \sum_{k=0}^{k_i-1}\sum_{t=1}^{m}\epsilon\left(\left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\|^2 + \left\|\mathbf{P}_i^{(k+1)} - \mathbf{P}_i^{(k)}\right\|^2\right).\end{aligned}$$

Taking the limit as $i \to +\infty$ in the above inequality and rearranging terms, we obtain

$$\begin{aligned}(\text{B.4})\quad &\frac{1}{2}\lambda_{\min}(L)\sum_{k=0}^{+\infty}\left\|\mathbf{W}^{(k+1)}-\mathbf{W}^{(k)}\right\|^2+\sum_{k=0}^{+\infty}\sum_{t=1}^{m}\epsilon\left(\left\|\mathbf{U}_i^{(k+1)}-\mathbf{U}_i^{(k)}\right\|^2+\left\|\mathbf{P}_i^{(k+1)}-\mathbf{P}_i^{(k)}\right\|^2\right)\\ &\leq \Phi\left(\mathbf{W}^{(0)},\mathbf{U}^{(0)},\mathbf{P}^{(0)},\mathbf{\Lambda}^{(0)}\right)-\Phi\left(\mathbf{W}^{*},\mathbf{U}^{*},\mathbf{P}^{*},\mathbf{\Lambda}^{*}\right)<\infty.\end{aligned}$$

Here the last inequality holds as our augmented Lagrangian function is unbounded only if our input is unbounded. Moreover, the sequence $\{(\mathbf{W}^{(k)},\mathbf{U}^{(k)},\mathbf{P}^{(k)},\mathbf{\Lambda}^{(k)})\}_{k=0}^{\infty}$ is bounded and we can complete the argument.

(B.4) implies that

$$\sum_{k=0}^{+\infty}\left\|\mathbf{W}^{(k+1)}-\mathbf{W}^{(k)}\right\|^2<\infty,\quad \sum_{k=0}^{+\infty}\sum_{i=1}^{m}\left\|\mathbf{U}_i^{(k+1)}-\mathbf{U}_i^{(k)}\right\|^2<\infty,\quad \sum_{k=0}^{+\infty}\sum_{i=1}^{m}\left\|\mathbf{P}_i^{(k+1)}-\mathbf{P}_i^{(k)}\right\|^2<\infty.$$

Hence, $\mathbf{W}^{(k+1)}-\mathbf{W}^{(k)}\to 0, \mathbf{U}^{(k+1)}-\mathbf{U}^{(k)}\to 0, \mathbf{P}^{(k+1)}-\mathbf{P}^{(k)}\to 0$. Due to the primal-dual relationship (B.3), we can thus conclude that $\mathbf{\Lambda}^{(k+1)}-\mathbf{\Lambda}^{(k)}\to 0$.

- *Step* 3*:* Derive a safeguard property.

Define the extended augmented Lagrangian function $G(\mathbf{W},\mathbf{U},\mathbf{P},\mathbf{\Lambda})=\Phi(\mathbf{W},\mathbf{U},\mathbf{P},\mathbf{\Lambda})+\sum_{i=1}^{m}g(\mathbf{U}_i)$. Recall the optimization optimality conditions for the ADMM updates (i.e., the $k+1$ iteration):

$$\begin{cases}0=h_i\left(\mathbf{U}_i^{(k+1)}-\mathbf{U}_i^{(k)}\right)+\mu_i\left(\mathbf{U}_i^{(k+1)}\mathbf{P}_i^{(k)}-\left(G\mathbf{W}^{(k+1)}\right)_i-\mathbf{\Lambda}_i^{(k)}\right)\mathbf{P}_i^{(k)\top}+\partial g\left(\mathbf{U}_i^{(k+1)}\right) & \forall i,\\ 0=w_i\nabla f\left(\mathbf{P}_i^{(k+1)}\right)-\mu_i\,\mathrm{symm}\left(\mathbf{U}_i^{(k+1)\top}\mathbf{\Lambda}_i^{(k+1)}\right) & \forall i,\\ \mathbf{\Lambda}_i^{(k+1)}=\mathbf{\Lambda}_i^{(k)}+\left(G\mathbf{W}^{(k+1)}\right)_i-\mathbf{U}_i^{(k+1)}\mathbf{P}_i^{(k+1)}.\end{cases}$$

Moreover, a stationary point satisfying $0\in\partial G(\mathbf{W}^{*},\mathbf{U}^{*},\mathbf{P}^{*},\mathbf{\Lambda}^{*})$ is equivalent to the KKT point property in (5.1). Subsequently, we want to bound the subgradient

$$\mathrm{dist}(0,\partial G(\mathbf{W}^{(k+1)},\mathbf{U}^{(k+1)},\mathbf{P}^{(k+1)},\mathbf{\Lambda}^{(k+1)}))$$

by the iterate difference, i.e., $\|\mathbf{P}^{(k+1)}-\mathbf{P}^{(k)}\|$, $\|\mathbf{U}^{(k+1)}-\mathbf{U}^{(k)}\|$, $\|\mathbf{W}^{(k+1)}-\mathbf{W}^{(k)}\|$,

$$\begin{aligned}&\mathrm{dist}\left(0,\partial G\left(\mathbf{W}^{(k+1)},\mathbf{U}^{(k+1)},\mathbf{P}^{(k+1)},\mathbf{\Lambda}^{(k+1)}\right)\right)\\ &\quad\leq\sum_{i=1}^{m}\left\|w_i\nabla f\left(\mathbf{P}_i^{(k+1)}\right)-\mu_i\,\mathrm{symm}\left(\mathbf{U}_i^{(k+1)\top}\mathbf{\Lambda}_i^{(k+1)}\right)\right\|\\ &\qquad+\sum_{i=1}^{m}\mathrm{dist}\left(0,\partial g\left(\mathbf{U}_i^{(k+1)}\right)-\mu_i\mathbf{\Lambda}_i^{(k+1)}\mathbf{P}_i^{(k+1)\top}\right)+\sum_{i=1}^{m}\left\|\left(G\mathbf{W}^{(k+1)}\right)_i-\mathbf{U}_i^{(k+1)}\mathbf{P}_i^{(k+1)}\right\|.\end{aligned}$$

We observe that the first term is 0, and the third term is identical to $\sum_{i=1}^{m} \|\mathbf{\Lambda}_i^{(k+1)} - \mathbf{\Lambda}_i^{(k)}\|$. It remains to bound the second term. Starting from the optimality condition w.r.t $\mathbf{U}_i^{(k+1)}$,

$$
\begin{aligned}
0 &= h_i \left(\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right) + \mu_i \left(\mathbf{U}_i^{(k+1)}\mathbf{P}_i^{(k)} - \left(G\mathbf{W}^{(k+1)}\right)_i - \mathbf{\Lambda}_i^{(k)}\right) {\mathbf{P}_i^{(k)}}^\top + \partial g\left(\mathbf{U}_i^{(k+1)}\right), \\
0 &= h_i \left(\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right) + \mu_i \left(\mathbf{\Lambda}_i^{(k+1)} + \mathbf{U}_i^{(k+1)}\left(\mathbf{P}_i^{(k)} - \mathbf{P}_i^{(k+1)}\right)\right) {\mathbf{P}_i^{(k)}}^\top + \partial g\left(\mathbf{U}_i^{(k+1)}\right), \\
0 &= h_i \left(\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right) + \mu_i \mathbf{\Lambda}_i^{(k+1)} \left({\mathbf{P}_i^{(k)}}^\top - {\mathbf{P}_i^{(k+1)}}^\top\right) \\
&\quad + \mu_i \mathbf{U}_i^{(k+1)} \left(\mathbf{P}_i^{(k)} - \mathbf{P}_i^{(k+1)}\right) {\mathbf{P}_i^{(k)}}^\top + \mu_i \mathbf{\Lambda}_i^{(k+1)} {\mathbf{P}_i^{(k+1)}}^\top + \partial g\left(\mathbf{U}_i^{(k+1)}\right).
\end{aligned}
$$

As $\{\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k+1)}\}_{k\geq 0}$ is bounded, i.e., step 1, there exists a constant D such that

$$
\left(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k+1)}\right) \in \mathcal{C},\ \mathcal{C} = \{(\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda}) | \|\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda}\| \leq D\}.
$$

Thus,

$$
\begin{aligned}
&\operatorname{dist}\left(0, \mu_i \mathbf{\Lambda}_i^{(k+1)} {\mathbf{P}_i^{(k+1)}}^\top + \partial g\left(\mathbf{U}_i^{(k+1)}\right)\right) \\
&\quad \leq h_i \left\|\mathbf{U}_i^{(k+1)} - \mathbf{U}_i^{(k)}\right\| + \mu_i \left(D + \|\mathbf{P}_i^{(k)}\|\right) \left\|\mathbf{P}_i^{(k)} - \mathbf{P}_i^{(k+1)}\right\|.
\end{aligned}
$$

Due to Lemma 5.3, we know that there exists a constant $\kappa > 0$ such that

$$
\begin{aligned}
&\operatorname{dist}\left(0, \partial G\left(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k+1)}\right)\right) \\
&\quad \leq \kappa \left(\|\mathbf{U}^{(k+1)} - \mathbf{U}^{(k)}\| + \|\mathbf{P}^{(k+1)} - \mathbf{P}^{(k)}\| + \|\mathbf{\Lambda}^{(k+1)} - \mathbf{\Lambda}^{(k)}\|\right).
\end{aligned}
$$

Based on Step 2, i.e., $\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)} \to 0, \mathbf{U}^{(k+1)} - \mathbf{U}^{(k)} \to 0, \mathbf{P}^{(k+1)} - \mathbf{P}^{(k)} \to 0$, there exists $d_{k+1} \in \partial G(\mathbf{W}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{P}^{(k+1)}, \mathbf{\Lambda}^{(k+1)})$ such that $\|d_{k+1}\| \to 0$. By the definition of general subgradient, we have $0 \in \partial G(\mathbf{W}^*, \mathbf{U}^*, \mathbf{P}^*, \mathbf{\Lambda}^*)$. Thus, any cluster point $(\mathbf{W}^*, \mathbf{U}^*, \mathbf{P}^*, \mathbf{\Lambda}^*)$ of a sequence $(\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)})$ generated by the ADMM is a stationary point, or KKT point equivalently.

- *Step* 4*:* Show that $G(\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda})$ is a Kurdyka–Łojasiewicz function.

Following the proof of Theorem 2.9 in [8], we can infer the global convergence of the sequence $\{\mathbf{W}^{(k)}, \mathbf{U}^{(k)}, \mathbf{P}^{(k)}, \mathbf{\Lambda}^{(k)}\}$ from the KŁ condition of the extended augmented Lagrangian function $G(\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda})$. Therefore, the final step is to prove that $G(\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda})$ is a Kurdyka–Łojasiewicz function.

Recall that

$$
G(\mathbf{W}, \mathbf{U}, \mathbf{P}, \mathbf{\Lambda}) = \sum_{i=1}^{m} w_i f(\mathbf{P}_i) + \sum_{i=1}^{m} \frac{\mu_i}{2} \left(\|(G\mathbf{W})_i - \mathbf{U}_i\mathbf{P}_i + \mathbf{\Lambda}_i\|^2 - \|\mathbf{\Lambda}_i\|^2\right) + \sum_{i=1}^{m} g(\mathbf{U}_i).
$$

The KŁ property is closed under summation [8]. Thus, we can check the above summands one by one. $\sum_{i=1}^{m} w_i f(\mathbf{P}_i)$ is strongly convex and hence satisfies the uniform convexity

property, and is a KŁ function [7, section 4.1]. $\sum_{i=1}^{m} \frac{\mu_i}{2} \left(\|(G\mathbf{W})_i - \mathbf{U}_i\mathbf{P}_i + \mathbf{\Lambda}_i\|^2 - \|\mathbf{\Lambda}_i\|^2 \right)$ is a polynomial function and thus semialgebraic, and semialgebraic functions satisfy the KŁ property [7, 8]. As $g(\cdot)$ is the indicator function over the special orthogonal group, it is a KŁ function (via Stiefel manifolds [7]).

This completes the proof of the theorem. ∎

Acknowledgments. We thank Alp Yurtsever and Suvrit Sra for discussing ADMM proofs of convergence with us. Credit for meshes used goes to [1, 4, 6, 9, 10, 11, 18, 23, 24, 41, 42, 45, 46, 47, 59, 61, 63, 68, 77, 80, 82, 83, 87, 88, 102, 106, 107, 114].

REFERENCES

[1] 4MULE8, *The Helm of Glencairn Mesh*, https://www.thingiverse.com/thing:1243621 (2016).
[2] N. Aigerman and Y. Lipman, *Injective and bounded distortion mappings in 3d*, ACM Trans. Graph., 32 (2013), 106.
[3] N. Aigerman, R. Poranne, and Y. Lipman, *Lifted bijections for low distortion surface mappings*, ACM Trans. Graph., 33 (2014), 69.
[4] AJade, *Pegasus for 28mm Tabletop Roleplaying Mesh*, https://www.thingiverse.com/thing:3955356 (2019).
[5] K. Akeley, J. D. Foley, D. F. Sklar, M. McGuire, J. F. Hughes, A. can Dam, and S. K. Feiner, *Computer Graphics – Principles and Practice*, Addison-Wesley, Upper Saddle River, NJ, 2013.
[6] N. Al-Badri and J. N. Nelles, *Nefertiti Mesh*, https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/ (2020).
[7] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, *Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-łojasiewicz inequality*, Math. Oper. Res., 35 (2010), pp. 438–457.
[8] H. Attouch, J. Bolte, and B. F. Svaiter, *Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods*, Math. Program., 137 (2013), pp. 91–129.
[9] I. Baran, *Stuffedtoy Mesh*, https://github.com/libigl/libigl-tutorial-data (2007).
[10] billyd, *Cat Stretch Mesh*, https://www.thingiverse.com/thing:1565405 (2016).
[11] Blender Foundation, *Blender – Free and Open Source 3D Creation Suite*, https://www.blender.org (2020).
[12] J. Bolte, A. Daniilidis, and A. Lewis, *The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems*, SIAM J. Optim., 17 (2007), pp. 1205–1223.
[13] D. Bommes, H. Zimmer, and L. Kobbelt, *Mixed-integer quadrangulation*, ACM Trans. Graph., 28 (2009), 77.
[14] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly, *Shape-up: Shaping discrete geometry with projections*, Comput. Graph. Forum, 31 (2012), pp. 1657–1667.
[15] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, *Projective dynamics: Fusing constraint projections for fast simulation*, ACM Trans. Graph., 33 (2014), 154.
[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2011), pp. 1–122.
[17] G. Brown and R. Narain, *WRAPD: Weighted rotation-aware ADMM for parametrization and deformation*, ACM Trans. Graph., 40 (2021), 82.
[18] Catiav5ftw, *Centrifugal Compressor Mesh*, https://www.thingiverse.com/thing:1165042 (2015).
[19] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, *A simple geometric model for elastic deformations*, ACM Trans. Graph. 29 (2010), 38.
[20] E. Chien, Z. Levi, and O. Weber, *Bounded distortion parametrization in the space of metrics*, ACM Trans. Graph., 35 (2016), 215.

[21] G. P. T. Choi and C. H. Rycroft, *Density-equalizing maps for simply connected open surfaces*, SIAM J. Imaging Sci., 11 (2018), pp. 1134–1178.
[22] S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon, *Isometry-aware preconditioning for mesh parameterization*, Comput. Graph. Forum, 36 (2017), pp. 37–47.
[23] colinfizgig, *Maltese Falcon Mesh*, https://www.thingiverse.com/thing:46631 (2013).
[24] K. Crane, *Blub mesh*, https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/ (2015).
[25] T. Davis, *SuiteSparse: A Sparse Matrix Algorithm Suite (Version* 5.8.1*)*, https://people.engr.tamu.edu/davis/welcome.html (2020).
[26] M. Desbrun, M. Meyer, and P. Alliez, *Intrinsic parameterizations of surface meshes*, Comput. Graph. Forum, 21 (2002), pp. 209–218.
[27] X. Du, N. Aigerman, Q. Zhou, S. Z. Kovalsky, Y. Yan, D. M. Kaufman, and T. Ju, *Lifting simplices to find injectivity*, ACM Trans. Graph., 39 (2020), 120.
[28] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, *Multiresolution analysis of arbitrary meshes*, In Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95, ACM, New York, pp. 173–182, 1995.
[29] D. Ezuz, J. Solomon, and M. Ben-Chen, *Reversible harmonic maps between discrete surfaces*, ACM Trans. Graph., 38 (2019), 15.
[30] Y. Fang, M. Li, C. Jiang, and D. M. Kaufman, *Guaranteed globally injective 3D deformation processing*, ACM Trans. Graph., 40 (2021), 75.
[31] L. P. Franca, *An algorithm to compute the square root of a* 3*x*3 *positive definite matrix*, Comput. Math. Appl., 18 (1989), pp. 459–466.
[32] X.-M. Fu, Y. Liu, and B. Guo, *Computing locally injective mappings by advanced MIPS*, ACM Trans. Graph., 34 (2015), 71.
[33] W. Gao, D. Goldfarb, and F. E. Curtis, *ADMM for multiaffine constrained optimization*, Optim. Methods Softw., 35 (2020), pp. 257–303.
[34] V. Garanzha, I. Kaporin, L. Kudryavtseva, F. Protais, N. Ray, and D. Sokolov, *Foldover-free maps in* 50 *lines of code*, ACM Trans. Graph., 40 (2021), 102.
[35] M. Gillespie, B. Springborn, and K. Crane, *Discrete conformal equivalence of polyhedral surfaces*, ACM Trans. Graph., 40 (2021), 103.
[36] J. C. Gower and G. B. Dijksterhuis, *Procrustes Problems*, Oxford Statist. Sci. Ser. 30. Oxford University Press, Oxford, 2004.
[37] X. Gu and S.-T. Yau, *Global conformal surface parameterization*, in Proceedings of the 2003 Eurographics symposium on Geometry Processing, ACM, New York, pp. 127–137, 2003.
[38] B. C. Hall, *Lie Groups, Lie Algebras, and Representations – An Elementary Introduction*. Springer, Cham, 2015.
[39] E. F. Hefetz, E. Chien, and O. Weber, *A subspace method for fast locally injective harmonic mapping*, Comput. Graph. Forum, 38 (2019), pp. 105–119.
[40] H. Hencky, *Über die form des elastizitätsgesetzes bei ideal elastischen stoffen*, Zeitschr. Tech. Phys., 9, 1928, pp. 215–220.
[41] J. Holinaty, *Brucewick Mesh*, https://github.com/odedstein/meshes/tree/master/objects/brucewick (2020).
[42] J. Holinaty, *Mushroom Mesh*, https://github.com/odedstein/meshes/tree/master/objects/mushroom (2020).
[43] M. Hong, Z.-Q. Luo, and M. Razaviyayn, *Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems*, SIAM J. Optim., 26 (2016), pp. 337–364.
[44] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo, *Tetrahedral meshing in the wild*, ACM Trans. Graph., 37 (2018), 60.
[45] hugoelec, *Goat* 5*k Yuanmingyuan* 12 *Zodiac Animals Mesh*, https://www.thingiverse.com/thing:42256 (2013).
[46] hugoelec, *Symmetrical Deer Head* 10*k Mesh*, https://www.thingiverse.com/thing:149354 (2013).
[47] A. Jacobson, *Algorithms and Interfaces for Real-Time Deformation of* 2*D and* 3*D Shapes*. PhD thesis, ETH Zürich, Zürich, 2013.
[48] A. Jacobson, *Geometry Processing - Parametrization (course)*, 2020.
[49] A. Jacobson et al, *libigl: A simple C++ geometry processing library*, https://libigl.github.io/ (2018).

[50] Z. Jiang, S. Schaefer, and D. Panozzo, *Simplicial complex augmentation framework for bijective maps*, ACM Trans. Graph., 36 (2017), 186.
[51] F. Kälberer, M. Nieser, and K. Polthier, *Quadcover - surface parameterization using branched coverings*, Comput. Graph. Forum, 26 (2007), pp. 375–384.
[52] S. Khashin, *Solution of Cubic and Quartic Equations C++*, manuscript.
[53] S. Z. Kovalsky, N. Aigerman, R. Basri, and Y. Lipman, *Large-scale bounded distortion mappings*, ACM Trans. Graph., 34 (2015), 191.
[54] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, *Least squares conformal maps for automatic texture atlas generation*, ACM Trans. Graph., 21 (2002), pp. 362–371.
[55] J. Li, A. M.-C. So, and W.-K. Ma, *Understanding notions of stationarity in nonsmooth optimization: A guided tour of various constructions of subdifferential for nonsmooth functions*, IEEE Signal Process. Mag., 37 (2020), pp. 18–31.
[56] M. Li, D. M. Kaufman, V. G. Kim, J. Solomon, and A. Sheffer, *OptCuts: Joint optimization of surface cuts and parameterization*, ACM Trans. Graph., 37 (2018), 247.
[57] Y. Lipman, *Bounded distortion mapping spaces for triangular meshes*, ACM Trans. Graph., 31 (2012), 108.
[58] Y. Lipman, *Bijective mappings of meshes with boundary and the degree in mesh processing*, SIAM J. Imaging Sci., 7 (2014), pp. 1263–1283.
[59] L. Liu, C. Ye, R. Ni, and X.-M. Fu, *Progressive parameterizations*, ACM Trans. Graph., 37 (2018), 41.
[60] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler, *A local/global approach to mesh parameterization*, in Proceedings of the Symposium on Geometry Processing, SGP '08, ACM, New York, 2008. pp. 1495–1504.
[61] M3DM, *Dead Tree - Tabletop Scater Terrain Mesh*, https://www.thingiverse.com/thing:4105303 (2020).
[62] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, and Y. Lipman, *Convolutional neural networks on surfaces via seamless toric covers*, ACM Trans. Graph., 36 (2017), 71.
[63] Max Planck Society e.V, *SMPL Human Body Mesh*, https://smpl.is.tue.mpg.de/en (2021).
[64] A. McAdams, A. Selle, R. Tamstorf, J. Teran, and E. Sifakis, *Computing the Singular Value Decomposition of 3x3 Matrices with Minimal Branching and Elementary Floating Point Operations, Technical report* 1690, University of Wisconsin-Madison, Department of Computer Sciences, Madison, Wisconsin, 2011.
[65] D. Minor, *Making space for cloth simulations using energy minimization*, in ACM SIGGRAPH 2018 Talks, ACM, New York, 2018, 41.
[66] D. Minor and D. Corral, *Smeat: ADMM based tools for character deformation*, in SIGGRAPH Asia 2018 Technical Briefs, ACM, New York, 2018, 2.
[67] P. Mullen, Y. Tong, P. Alliez, and M. Desbrun, *Spectral conformal parameterization*, Comput. Graph. Forum, 27 (2008), pp. 1487–1494.
[68] MustangDave, *Berry Bear Mesh*, https://www.thingiverse.com/thing:1161576 (2015).
[69] A. Myles, N. Pietroni, and D. Zorin, *Robust field-aligned global parametrization*, ACM Trans. Graph., 33 (2014), 135.
[70] Alexander N., E. Saucan, and Y. Y. Zeevi, *Geometry-based distortion measures for space deformation*, Graph. Models, 100 (2018), pp. 12–25.
[71] P. Neff, B. Eidel, and R. J. Martin, *Geometry of logarithmic strain measures in solid mechanics*, Arch. Ration. Mech. Anal., 222 (2016), pp. 507–572.
[72] Yurii N., *Lectures on Convex Optimization*, Springer Optim. Appl. 137, Springer, Cham, Switzerland, 2018.
[73] X. Nian and F. Chen, *Planar domain parameterization for isogeometric analysis based on Teichmüller mapping*, Comput. Methods Appl. Mech. Engi., 311 (2016), pp. 41–55.
[74] W. Ouyang, Y. Peng, Y. Yao, J. Zhang, and B. Deng, *Anderson acceleration for nonconvex ADMM based on Douglas-Rachford splitting*, Comput. Graph. Forum, 39 (2020), pp. 221–239.
[75] M. Overby, G. E. Brown, J. Li, and R. Narain, *ADMM ⊇ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints*, IEEE Trans. Vis. Comput. Graph., 23 (2017), pp. 2222–2234.

[76] M. Overby, D. Kaufman, and N. Rahul, *Globally injective geometry optimization with non-injective steps*, Comput. Graph. Forum, 40 (2021), pp. 111–123.
[77] paBlury, *Small Cactus with Pot Mesh*, https://www.thingiverse.com/thing:2720011 (2017).
[78] K. Pal, C. Schüller, D. Panozzo, O. Sorkine-Hornung, and T. Weyrich, *Content-aware surface parameterization for interactive restoration of historical documents*, Comput. Graph. Forum, 33 (2014), pp. 401–409.
[79] Z. Pan, H. Bao, and J. Huang, *Subspace dynamic simulation using rotation-strain coordinates*, ACM Trans. Graph., 34 (2015), 242.
[80] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross, *Shape modeling with point-sampled geometry (3D octopus mesh)*, ACM Trans. Graph., 22 (2003), pp. 641–650.
[81] U. Pinkall and K. Polthier, *Computing discrete minimal surfaces and their conjugates*, Exp. Math., 2 (1993), pp. 15–36.
[82] Publicdomainvectors.org, *Blue Robot Image*, https://publicdomainvectors.org/en/free-clipart/Blue-robot/62916.html (2017).
[83] Publicdomainvectors.org. *Glossy Red Lips Image*, https://publicdomainvectors.org/en/free-clipart/Glossy-red-lips/80649.html (2019).
[84] M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung, Scalable locally injective mappings. *ACM Trans. Graph.*, 36 (2017), 18.
[85] R. Sanyal, Sk. M. Ahmed, M. Jaiswal, and K. N. Chaudhury, *A scalable ADMM algorithm for rigid registration*, IEEE Signal Process. Lett., 24 (2017), pp. 1453–1457.
[86] R. Sawhney and K. Crane, *Boundary first flattening*, ACM Trans. Graph., 37 (2017), 5.
[87] schlossbauer, *Brain Ooze Mesh*, https://www.thingiverse.com/thing:4169343 (2020).
[88] schlossbauer, *Slime Mold Mesh*, https://www.thingiverse.com/thing:4756744 (2021).
[89] P. Schmidt, J. Born, M. Campen, and L. Kobbelt, *Distortion-minimizing injective maps between surfaces*, ACM Trans. Graph., 38 (2019), 156.
[90] P. Schmidt, M. Campen, J. Born, and L. Kobbelt, *Inter-surface maps via constant-curvature metrics*, ACM Trans. Graph., 39 (2020), 119.
[91] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, *Inter-surface mapping*, ACM Trans. Graph., 23 (2004), pp. 870–877.
[92] S. Sellán, N. Aigerman, and A. Jacobson, *Developability of heightfields via rank minimization*, ACM Trans. Graph., 39 (2020), 109.
[93] N. Sharp and K. Crane, *Variational surface cutting*, ACM Trans. Graph., 37 (2018), 156.
[94] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, *ABF++ : Fast and robust angle based flattening*, ACM Trans. Graph., 24 (2004), pp. 311–330.
[95] H. Shen, Z. Jiang, D. Zorin, and D. Panozzo, *Progressive embedding*, ACM Trans. Graph., 38 (2019), 32.
[96] H. Si, *Tetgen, a Delaunay-based quality tetrahedral mesh generator*, ACM Trans. Math. Softw., 41 (2015), 11.
[97] B. Smith, F. De Goes, and T. Kim, *Analytic eigensystems for isotropic distortion energies*, ACM Trans. Graph., 38 (2019), 3.
[98] J. Smith and S. Schaefer, *Bijective parameterization with free boundaries*, ACM Trans. Graph., 34 (2015), 70.
[99] Y. Soliman, D. Slepčev, and K. Crane, *Optimal cone singularities for conformal flattening*, ACM Trans. Graph., 37 (2018), 105.
[100] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher, *Earth mover's distances on discrete surfaces*, ACM Trans. Graph., 33 (2014), 67.
[101] O. Sorkine and M. Alexa, *As-rigid-as-possible surface modeling*, in Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07, A.K. Peters, Wellesleg, MA, 2007, pp. 109–116.
[102] O. Sorkine and D. Cohen-Or, *Camelhead Mesh*, https://github.com/libigl/libigl-tutorial-data (2004).
[103] B. Springborn, P. Schröder, and U. Pinkall, *Conformal equivalence of triangle meshes*, ACM Trans. Graph., 27 (2008), pp. 1–11.

[104] J.-P. Su, X.-M. Fu, and L. Liu, *Practical foldover-free volumetric mapping construction*, Comput. Graph. Forum, 38 (2019), pp. 287–297.
[105] J.-P. Su, C. Ye, L. Liu, and X.-M. Fu, *Efficient bijective parameterizations*, ACM Trans. Graph., 39 (2020), 111.
[106] *The Stanford 3D Scanning Repository*, Bunny and Armadillo Meshes. http://graphics.stanford.edu/data/3Dscanrep/ (2020).
[107] Toawi, *Bread – Robinson Crusoe Mesh*, https://www.thingiverse.com/thing:4414133 (2020).
[108] W. T. Tutte, *How to draw a graph*, Proc. London Math. Soc.(3), 13 (1963), pp. 743–767.
[109] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, *Painless stochastic gradient: Interpolation, line-search, and convergence rates*, in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, ed., Curran Associates, Red Hook, NY, 2019, pp. 3727–3740.
[110] J. Wang and L. Zhao, *Nonconvex generalization of alternating direction method of multipliers for nonlinear equality constrained problems*, Results Control Optim., 2 (2021), 100009.
[111] Y. Wang, Y. Wotai, and J. Zheng, *Global convergence of ADMM in nonconvex nonsmooth optimization*, J. Sci. Comput., 78 (2019), pp. 29–63.
[112] O. Weber and D. Zorin, *Locally injective parametrization with arbitrary fixed boundaries*, ACM Trans. Graph., 33 (2014), 75.
[113] Z. Xu, M. A. T. Figueiredo, X. Yuan, C. Studer, and T. Goldstein, *Adaptive relaxed ADMM: Convergence theory and practical implementation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, 2017, pp. 7389–7398.
[114] YahooJAPAN, *Monkey Mesh*, https://www.thingiverse.com/thing:182232 (2013).
[115] L. Yang, T. K. Pong, and X. Chen, *Alternating direction method of multipliers for a class of nonconvex and nonsmooth problems with applications to background/foreground extraction*, SIAM J. Imaging Sci., 10 (2017), pp. 74–110.
[116] M.-H. Yueh, T. Li, W.-W. Lin, and S.-T. Yau, *A novel algorithm for volume-preserving parameterizations of 3-manifolds*, SIAM J. Imaging Sci., 12 (2019), pp. 1071–1098.
[117] J. Zhang, Y. Duan, Y. Lu, M. K. Ng, and H. Chang, *Bilinear constraint based ADMM for mixed Poisson-Gaussian noise removal*, Inverse Probl. Imaging, 15 (2021), pp. 339–366.
[118] J. Zhang, S. Ma, and S. Zhang, *Primal-dual optimization algorithms over Riemannian manifolds: An iteration complexity analysis*, Math. Program., 184 (2020), pp. 445–490.
[119] J. Zhang, Y. Peng, W. Ouyang, and B. Deng, *Accelerating ADMM for efficient simulation and optimization*, preprint, arXiv:1909.00470, 2019.
[120] T. Zhang and Z. Shen, *A fundamental proof of convergence of alternating direction method of multipliers for weakly convex optimization*, J. Inequal. Appl., 2019 (2019), pp. 1–21.
[121] Y. Zhu, R. Bridson, and D. M. Kaufman, *Blended cured quasi-Newton for distortion optimization*, ACM Trans. Graph., 37 (2018), 40.