Geometric Optimization via Composite Majorization

ANNA SHTENGEL, Weizmann Institute of Science
ROI PORANNE and OLGA SORKINE-HORNUNG, ETH Zurich
SHAHAR Z. KOVALSKY, Duke University
YARON LIPMAN, Weizmann Institute of Science

Many algorithms on meshes require the minimization of composite objectives, *i.e.*, energies that are compositions of simpler parts. Canonical examples include mesh parameterization and deformation. We propose a second order optimization approach that exploits this composite structure to efficiently converge to a local minimum.

Our main observation is that a convex-concave decomposition of the energy constituents is simple and readily available in many cases of practical relevance in graphics. We utilize such convex-concave decompositions to define a tight convex majorizer of the energy, which we employ as a *convex* second order approximation of the objective function. In contrast to existing approaches that largely use only local convexification, our method is able to take advantage of a more global view on the energy landscape. Our experiments on triangular meshes demonstrate that our approach outperforms the state of the art on standard problems in geometry processing, and potentially provide a unified framework for developing efficient geometric optimization algorithms.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; • **Theory of computation** → **Nonconvex optimization**;

Additional Key Words and Phrases: second order optimization, majorization, convex-concave, distortion, geometry processing

ACM Reference format:
Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Trans. Graph.* 36, 4, Article 38 (July 2017), 11 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073618

1 INTRODUCTION

Mesh processing tasks in computer graphics, including deformation and parameterization, are often cast as nonlinear minimization problems of the general form:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{1}$$

Active research effort is dedicated to develop numerical optimization methods for the solution of such problems, taking advantage of the known context, *e.g.*, the geometric structure of the problem.

Existing optimization algorithms typically produce a sequence of approximations, $\mathbf{x}_n$, designed to converge to a solution of (1). To this end, most approaches use, either explicitly or implicitly, a local quadratic approximation of the objective function: they construct an osculating convex quadric to f at $\mathbf{x}_n$, whose minimization determines the next approximation $\mathbf{x}_{n+1}$. From this point of view, the difference between the various methods lies in the particular choice of an osculating quadric, or more precisely, the choice of its *Hessian*.

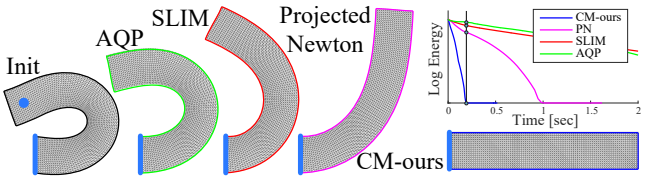


Fig. 1. Deforming a bar using different methods for minimizing the symmetric ARAP energy. In this example, we first deform the bar into a bent position (left); constrained points are highlighted in blue. We then measure the energy as we release the bar and let it reach its rest pose. The figure shows a snapshot of the state each method achieved at the time marked on the graph. Our Composite Majorization (CM-ours) converges faster than Projected Newton [Teran et al. 2005], SLIM [Rabinovich et al. 2017] and AQP [Kovalsky et al. 2016].

The archetypal Newton algorithm [Lange 2013; Nocedal and Wright 2006] uses the Hessian of f itself to define the osculating quadric. For strictly convex functions, this leads to a well-defined algorithm with quadratic order of convergence. For non-convex functions, however, the Hessian is often indefinite and thus Newton's algorithm is no longer guaranteed to be a descent algorithm. Several general heuristics exist on how to modify Newton's Hessian so as to force it into being positive semidefinite. Unfortunately, there is no clear generic strategy for such modification that works well in all, or even most cases. Consequently, generic algorithms are usually satisfied with just finding *some* decent direction, and Newton's desirable convergence rates are often not attained in practice.

In computer graphics, a specific effort is dedicated to developing efficient optimization algorithms for objective functions defined on meshes. These algorithms take advantage of the particular structure of the energies used in geometry processing and introduce, albeit sometimes implicitly, a *convex* osculating quadric used to determine their iterations. For instance, [Kovalsky et al. 2016; Liu et al. 2008; Sorkine and Alexa 2007] essentially replace the Hessian with the mesh Laplacian, [Liu et al. 2016] further use low-rank quasi-Newton updates to better approximate the Hessian, and [Rabinovich et al. 2017] reweigh the Laplacian to improve the effectiveness of their iterations. These are all *first order* methods, meaning they do not directly use second order derivatives of the energy, and therefore generally fail to achieve high convergence rate, particularly as they approach convergence.

Our goal is to devise a second order optimization approach applicable to a generic class of composite nonlinear energies in computer graphics; namely, we are concerned with objective functions that can be represented as the composition of simpler functions. Our

DOI: http://dx.doi.org/10.1145/3072959.3073618

strategy for picking a convex osculating quadric at $\mathbf{x}_n$ is based on: (i) exploiting the composite structure for constructing a *convex majorizer* to f centered at $\mathbf{x}_n$, and (ii) computing its Hessian at $\mathbf{x}_n$. The majorizer provides a tight convex upper bound to f in a well understood, nontrivial neighborhood of $\mathbf{x}_n$ and therefore provides a well justified choice of a positive semidefinite Hessian at $\mathbf{x}_n$. This eliminates the need to heuristically enforce positive semidefinitness.

Although a tight convex majorizer for a general non-linear function f is often difficult to obtain, we make the observation that it can be derived analytically from convex-concave decompositions of the functions composing f. Consequently, the resulting Hessian formula at $\mathbf{x}_n$ takes a simple analytic form and is efficient to compute in practice.

We argue that common problems in computer graphics are often compatible with our assumptions on the composite structure of the objective function. We demonstrate the utility of our approach in geometry processing of triangular surface meshes via instantiation of our Hessian formula for several popular energies, such as energies defined in terms of singular values. In general, applying our method requires convex-concave decompositions of the functions composing f. We evaluate the performance of the proposed approach and show that it outperforms state of the art algorithms on a variety of geometric optimization problems. Figure 1 shows deformations of a bar released from a bent initial configuration (left) computed using several state of the art methods, further details are provided in Section 5; our method, Composite Majorization (CM-ours), is the first to converge to the rest state of the bar (right).

2 BACKGROUND

A general meta-algorithm for the unconstrained minimization of an energy f is summarized in Algorithm 1. It iteratively solves

$$H\mathbf{p} = -\nabla f(\mathbf{x}), \tag{2}$$

where H is a positive definite matrix and $\mathbf{x}$ is the current iterate, to determine a search direction $\mathbf{p}$, and takes a step in the direction $\mathbf{p}$ that reduces the value of f. To simplify the exposition, we describe the unconstrained case; constraints, which are integral and important to many problems, can be handled via the solution of the corresponding KKT system in the linear case [Nocedal and Wright 2006] or by adding penalty to the objective function in the non-linear case, see Section 5.3 for an example.

Since H is positive definite, the solution of $H\mathbf{p} = -\nabla f(\mathbf{x})$ corresponds to a minimum of the *convex* osculating quadric to f at $\mathbf{x}$, given by

$$q(\mathbf{z}) = \frac{1}{2}(\mathbf{z}-\mathbf{x})^{\mathsf{T}} H (\mathbf{z}-\mathbf{x}) + \nabla f(\mathbf{x})^{\mathsf{T}} (\mathbf{z}-\mathbf{x}). \tag{3}$$

Convexity is essential, as it guarantees that the search direction $\mathbf{p}$ is indeed a descent direction, *i.e.*, $\mathbf{p}^{\mathsf{T}}\nabla f(\mathbf{x}) < 0$. Line search then ensures that each iteration of Algorithm 1 reduces the objective.

From this perspective, nonlinear optimization algorithms differ in the choice of Hessian H of the osculating quadric they employ, as well as how they enforce its positive definiteness. We present our approach for choosing H in Section 3. The rest of this section is devoted to an overview of related methods and puts our approach in a relevant context.

Algorithm 1: Meta-algorithm for nonlinear optimization

Input: initial guess $\mathbf{x}$
repeat
  $\mathbf{p} \leftarrow -H^{-1}\nabla f(\mathbf{x})$; (compute search direction)
  $t \leftarrow \arg\min_{\tau\in(0,1]} f(\mathbf{x}+\tau\mathbf{p})$; (perform line search)
  $\mathbf{x} \rightarrow \mathbf{x} + t\,\mathbf{p}$; (make a step)
until *convergence*;

Newton's algorithm. Newton's algorithm makes the choice $H = \nabla^2 f$. This amounts to having the quadric q coincide with f up to second order. For convex functions this implies that $\mathbf{p}$ is a descent direction. Furthermore, under additional mild assumptions, Algorithm 1 with this choice of osculating quadric converges quadratically – probably Netwon algorithm's most attractive property.

However, when optimizing a non-convex function, which is the typical case in geometry processing, $\nabla^2 f$ is not positive semidefinite (PSD) and $\mathbf{p}$ is no longer guaranteed to be a descent direction. Generic techniques simply replace $\nabla^2 f$ with a positive definite approximating matrix H. The simplest techniques, such as modifying the true Hessian by adding a multiple of the identity, are usually suboptimal and sacrifice convergence rate. Other heuristics, such as modified factorizations (*e.g.*, spectral, Cholesky) and projection may perform better, but they also introduce a significant computational overhead and may become computationally prohibitive. Unfortunately, there is no universal answer to what constitutes a good approximation. See Section 3.4 in [Nocedal and Wright 2006] for a comprehensive discussion on Newton's method with Hessian modifications.

Hessian modifications have also been adapted to geometry processing in several works [Fu and Liu 2016; Teran et al. 2005]. These works observe that in the common case of separable energies, which decompose as a sum over the elements of the mesh, it is sufficient to modify the sub-Hessians corresponding to each mesh element. Teran et al. [2005] propose to individually project the Hessian of each element onto the PSD cone, thus alleviating the computational burden of projecting the entire, usually large, Hessian at once. The resulting approximate Hessian is PSD due to linearity, and has been observed to work better in practice than the generic Hessian modifications [Liu et al. 2016].

Fig. 2 compares different Hessian modification methods for minimizing the symmetric Dirichlet energy (see equation (24)) of a small-scale sphere mesh comprising $\approx$ 400 vertices: Newton's method (without any modification), our method (CM-ours), the per element Hessians projections (PN), and full Hessian projections (Full PN). For the Full PN we tested two variants: (i) replacing eigenvalues of the Hessian that are smaller than ε to ε (Full-PN(a)); or (ii) adding a multiple of the identity matrix to the Hessian (Full-PN(b)). As can be seen from the graph, the latter "off-the-shelf" projections lead to longer damped phases. Note that the standard Newton's method fails to find a descent direction at some point and thus halts.

First order methods. Motivated by the particular structure of the energies commonly used in geometry processing, several works have exploited the mesh Laplacian [Pinkall and Polthier 1993] for approximating the Hessian. The Laplacian is implicitly used in the

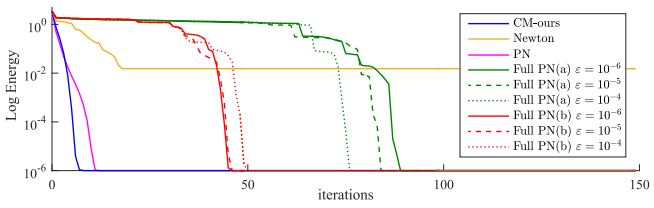


Fig. 2. Energy plots for parameterizing a small sphere mesh using different Hessian modifications.

local/global algorithm [Liu et al. 2008; Sorkine and Alexa 2007] for minimizing the as-rigid-as-possible energy. Kovalsky et al. [2016] advocate using the Laplacian instead of the true Hessian for general energies in geometry processing. Liu et al. [2016] suggest to refine the Hessian approximation for elastic simulations by applying low-rank quasi-Newton updates to the Laplacian. These works all benefit from the observation that the Laplacian is, by definition, positive semidefinite. Furthermore, by exploiting the fixed Hessian approximation (*i.e.*, independent of $\mathbf{x}$) they are able to devise particularly efficient iterations.

Rabinovich et al. [2017] are inspired by the fast initial progress of global-local iterations and advocate the use of a reweighted Laplacian for the minimization of nonlinear energies on meshes. Their algorithm is shown to be very efficient in minimizing the energy from arbitrary feasible initializations, but it significantly slows down near an optimum, and so additional Newton iterations might be required if an accurate solution is pursued.

Generally, since first order methods do not fully exploit second order information, they usually do not achieve the convergence rates associated with Newton-type algorithms. This is particularly the case when approaching a minimum, where Newton methods enter their powerful, quadratically convergent phase [Boyd and Vandenberghe 2004].

Gauss-Newton. The Gauss-Newton algorithm is often discussed in the context of nonlinear least squares [Boyd and Vandenberghe 2004; Nocedal and Wright 2006]. It considers energies of the form $f = \frac{1}{2}\sum {g_i}^2$, for which the Hessian is given by

$$\nabla^2 f = \sum_i \nabla g_i^\mathsf{T} \nabla g_i + \sum_i g_i \nabla^2 g_i \,.$$

The Gauss-Newton method omits the last term of $\nabla^2 f$ and instead uses

$$H = \sum_i \nabla g_i^\mathsf{T} \nabla g_i,$$

which is clearly PSD. The rationale is that when the g_i values are small (*e.g.*, close to a minimum), H well approximates the Hessian $\nabla^2 f$. Gauss-Newton has been successfully used in computer graphics for problems of adequate form, *e.g.*, [Huang et al. 2009; Tang et al. 2014].

Functions of the more general form $f = \sum_i h(g_i)$ have also been studied in [Martens 2010; Schraudolph 2002]. Their generalized Gauss-Newton Hessian approximation

$$H = \sum_i \nabla g_i^\mathsf{T} (\nabla^2 h) \nabla g_i \tag{4}$$

is derived similarly to the Gauss-Newton approximation, and reduces to it for $h(u) = \frac{1}{2}u^2$. It is guaranteed to be PSD if h is convex.

Majorization-Minimization and Convex-concave. Majorization Minimization (MM) is an optimization framework that works by fitting, at each iteration, a convex surrogate upper bound, called a *majorizer*, which is minimized to drive the objective function downhill. It thus replaces a difficult optimization with a sequence of simpler optimization problems. Simplicity is often attained by variable separation, linearization, convexification and other approaches that exploit the structure of specific problems, see [Hunter and Lange 2004] and [Lange 2013, chapter 8].

The use of MM is prevalent in various fields of numerical optimization, albeit often implicitly: in statistics, the popular Expectation Maximization (EM) is a special instance of MM [Lange 2013, chapter 9]; so are the classes of iteratively reweighted least squares (IRLS) and iterative shrinkage thresholding (IST) algorithms, often considered in the literature of sparse linear inverse problems [Daubechies et al. 2004, 2010], or Stress Majorization for multidimensional scaling [Gansner et al. 2004], to name just a few examples. Instances of MM have also been employed in computer graphics, for example in texture synthesis [Kwatra et al. 2005] or feature matching [Lipman et al. 2014]. In fact, the popular local-global procedure for the minimization of the as-rigid-as-possible energy [Liu et al. 2008; Sorkine and Alexa 2007] is another example of a MM algorithm.

Most related to this work is the convex-concave procedure [Lipp and Boyd 2014; Yuille and Rangarajan 2003]. It is a subfamily of MM algorithms concerned with objective functions f that decompose as the sum $f = f^+ + f^-$ of a convex and a concave function. Given such decomposition, a global convex majorizer of f is easily constructed by replacing the concave f^- with its linear approximation at $\mathbf{x}_n$. As with the more general MM, this approach has also (implicitly) found applications in computer graphics; for example, Yuille and Rangarajan [2003] show that Sinkhorn's algorithm, used by Solomon et al., [2015] for the computation of geometric optimal transportation, is an instance of the convex-concave procedure.

In the context of Algorithm 1 and equation (2), it is thus natural to set the approximate Hessian to be the Hessian of the convex majorizer,

$$H = \nabla^2 f^+, \tag{5}$$

which is PSD by construction; this approach is also known as the MM-Gradient algorithm, see [Lange 2013, chapter 10.4]. Nevertheless, while the majority of functions admit a convex-concave decomposition [Hartman et al. 1959], it is not unique, and the specific chosen decomposition directly affects H and thus the convergence. Finding efficient convex-concave decompositions is typically difficult for complicated functions and, in general, remains a challenge.

3 APPROACH

This paper is concerned with the choice of a *convex* approximate Hessian H to be used in the generic nonlinear optimization Algorithm 1. We propose a general methodology for choosing H in a broad class of non-convex optimization problems, demonstrated to be particularly suitable for geometry processing tasks.

Our approach is based on a novel convex majorization technique for composition of functions for which convex-concave decompositions are known. In turn, we use the majorizer's Hessian as H in Algorithm 1. Our choice of H is therefore: (i) guaranteed to be PSD; (ii) corresponds to a non-trivial local convex majorizer; (iii) often takes simple analytic form and thus is simple to evaluate; and (iv) empirically outperforms alternative choices.

We begin by highlighting our main results and subsequently present their rigorous definitions and detailed derivations.

3.1 Summary of main results

We consider energies of the form

$$f(\mathbf{x}) = \sum_i h_i(\mathbf{g}_i(\mathbf{x})) = \sum_i \left(h_i \circ \mathbf{g}_i\right)(\mathbf{x}), \tag{6}$$

where $h_i : \mathbb{R}^k \to \mathbb{R}$ and $\mathbf{g}_i : \mathbb{R}^n \to \mathbb{R}^k$ are C^2 functions with *convex-concave decompositions*. That is, they each decompose as

$$h_i = h_i^+ + h_i^-, \qquad \mathbf{g}_i = \mathbf{g}_i^+ + \mathbf{g}_i^-, \tag{7}$$

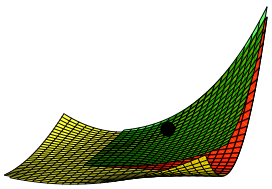
with h_i^+ *convex* and the vector-valued functions $\mathbf{g}_i^+$ *convex* in each of their coordinates and, respectively, h_i^- and $\mathbf{g}_i^-$ *concave*.

We will first show how to use these decompositions to construct a *convex majorizer* to f centered at a general point $\mathbf{x}_0$ that takes the form

$$\bar{f} = \sum_i \bar{h}_i \circ \left[\mathbf{g}_i\right], \tag{8}$$

where $\bar{h}_i$ is a majorizer of h_i centered at $\mathbf{g}_i(\mathbf{x}_0)$, and $\left[\mathbf{g}_i\right]$ is a vector function whose entries are either majorizers or minorizers of $\mathbf{g}_i$ centered at $\mathbf{x}_0$.

We further prove that $\bar{f}$ is a *non-trivial local majorizer*, in the sense that we define the non-trivial set $\Omega(\mathbf{x}_0) \subseteq \mathbb{R}^n$ for which $\bar{f}$ majorizes f. This comes in contrast to an arbitrarily defined local majorizer, that may majorize f in only an arbitrarily small neighborhood of $\mathbf{x}_0$. The inset illustrates our majorizer (green) for a simple 2-dimensional composed function (yellow) centered at the black disk; the region $\Omega(\mathbf{x}_0)$, over which majorization is guaranteed, is colored red.



Lastly, since $\bar{f}$ is a convex majorizer, it is natural to employ its minimization as a proxy for the minimization of f itself. Therefore, we propose to set H in Algorithm 1 to be the *positive semidefinite Hessian* of $\bar{f}$ at $\mathbf{x}_0$, given by

$$\begin{aligned} H = \nabla^2 \bar{f} = &\sum_i \left[\frac{\partial \mathbf{g}_i}{\partial \mathbf{x}}\right]^\mathsf{T} \nabla^2 h_i^+ \left[\frac{\partial \mathbf{g}_i}{\partial \mathbf{x}}\right] \\ &+ \sum_{i,j} \left(\frac{\partial h_i}{\partial u_j}\right)_+ \nabla^2 g_{ij}^+ \;+\; \sum_{i,j} \left(\frac{\partial h_i}{\partial u_j}\right)_- \nabla^2 g_{ij}^- \end{aligned} \tag{9}$$

where $(\cdot)_+$ keeps only positive numbers (linear rectifier), $(\cdot)_-$ only negative numbers, and g_{ij}^+ and g_{ij}^- are the j-th coordinates of the vector-valued functions $\mathbf{g}_i^+$ and $\mathbf{g}_i^-$, respectively.

Since H only involves the Hessian of the convex-concave parts of the h_i's and $\mathbf{g}_i$'s, it is simple to compute. In fact, it often takes an analytic form, as we demonstrate in Section 4. Moreover, we show in practice that this choice of H typically outperforms alternative approaches for choosing a convex approximate Hessian.

3.2 Derivation of majorizer and Hessian

To simplify the notations, let us omit the summation in equation (6) and consider a function of the form

$$f(\mathbf{x}) = h(\mathbf{g}(\mathbf{x})) = h(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})) \tag{10}$$

and let $h = h^+ + h^-$ and $g_j = g_j^+ + g_j^-$ be the respective convex-concave decompositions. Our derivations and results straightforwardly extend to the form (6).

Preliminaries. For a scalar function r with a convex-concave decomposition $r = r^+ + r^-$ we define its *majorizer* at $\mathbf{x}_0$ as

$$\bar{r}(\mathbf{x}; \mathbf{x}_0) = r^+(\mathbf{x}) + r^-(\mathbf{x}_0) + \nabla r^-(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0), \tag{11}$$

and its *minorizer* at $\mathbf{x}_0$ as

$$\underline{r}(\mathbf{x}; \mathbf{x}_0) = r^-(\mathbf{x}) + r^+(\mathbf{x}_0) + \nabla r^+(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0). \tag{12}$$

Note that $\bar{r}$ ($\underline{r}$) satisfies:

(1) It is a convex (concave) function.
(2) It coincides with r up to first order (*i.e.*, value and first derivative) at $\mathbf{x} = \mathbf{x}_0$.
(3) It is a global majorizer (minorizer) of r. That is, for all $\mathbf{x}$

$$\underline{r}(\mathbf{x}; \mathbf{x}_0) \le r(\mathbf{x}) \le \bar{r}(\mathbf{x}; \mathbf{x}_0).$$

Derivation. Fix $\mathbf{x}_0$, set $\mathbf{u}_0 = \mathbf{g}(\mathbf{x}_0)$ and let

$$s_j(\mathbf{u}) = \operatorname{sign}\left(\frac{\partial \bar{h}}{\partial u_j}(\mathbf{u}; \mathbf{u}_0)\right),$$

where we define here the sign function via

$$\operatorname{sign}(t) = \begin{cases} 1 & t \ge 0 \\ -1 & t < 0 \end{cases} \tag{13}$$

Intuitively, s_j is determined by whether the majorizer of h at $\mathbf{u}_0$ is increasing- or decreasing-monotone in its j-th entry.

Next, define the function $[\mathbf{g}] : \mathbb{R}^n \to \mathbb{R}^k$ by each of its entries

$$\left[g_j\right](\mathbf{x}; \mathbf{x}_0) = \begin{cases} \bar{g}_j(\mathbf{x}; \mathbf{x}_0) & s_j(\mathbf{u}_0) > 0 \\ \underline{g}_j(\mathbf{x}; \mathbf{x}_0) & s_j(\mathbf{u}_0) < 0 \end{cases} \tag{14}$$

Note that each entry of $[\mathbf{g}]$ is either a majorizer or a minorizer of the corresponding entry in $\mathbf{g}$ at $\mathbf{u}_0$, in correspondence with the monotonicity pattern of $\bar{h}$.

Equipped with these definitions we now propose our majorizer for f at $\mathbf{x}_0$, and define

$$\bar{f}(\mathbf{x}; \mathbf{x}_0) = \bar{h}\left([\mathbf{g}](\mathbf{x}; \mathbf{x}_0) \;;\; \mathbf{u}_0\right), \tag{15}$$

thus concretizing the definition of equation (8).

To show that $\bar{f}$ is a majorizer, we first have to define the domain on which it holds. We define

$$S(\mathbf{x}_0) = \left\{\mathbf{x} \mid s_j([\mathbf{g}](\mathbf{x}; \mathbf{x}_0)) = s_j(\mathbf{u}_0)\right\},$$

the set of all points $\mathbf{x}$ for which $[\mathbf{g}]$ is compatible with the monotonicity pattern of $\bar{h}$ at $\mathbf{u}_0$. Our majorizing domain $\Omega(\mathbf{x}_0)$ is defined by restricting our attention to the path connected component subset containing $\mathbf{x}_0$:

$$\Omega(\mathbf{x}_0) = \{\mathbf{x} \mid \mathbf{x} \text{ is path connected to } \mathbf{x}_0 \text{ in } S(\mathbf{x}_0)\}. \tag{16}$$

The inset figure in Section 3.1 depicts an example of a majorizer of a 2-dimensional function f over the domain $\Omega(\mathbf{x}_0)$ marked red over the graph of f.

Proposition 3.1. *The function $\bar{f}$ defined in equation* (15) *is a convex majorizer of f at $\mathbf{x}_0$ over the set $\Omega(\mathbf{x}_0)$ defined in equation* (16).

Proof. First, to see that $\bar{f}$ is convex over $\Omega(\mathbf{x}_0)$ let us consider its Hessian. Applying the chain rule yields

$$\nabla^2_{\mathbf{x}} \bar{f}(\mathbf{x};\mathbf{x}_0) = \frac{\partial [\mathbf{g}]}{\partial \mathbf{x}}^{\mathsf{T}} \nabla^2 h^+ \frac{\partial [\mathbf{g}]}{\partial \mathbf{x}} + \sum_j \frac{\partial \bar{h}}{\partial u_j} \begin{cases} \nabla^2 g_j^+ & s_j(\mathbf{u}_0) \geq 0 \\ \nabla^2 g_j^- & s_j(\mathbf{u}_0) < 0 \end{cases}. \tag{17}$$

Note that the first term is always PSD. For $\mathbf{x} \in \Omega(\mathbf{x}_0)$ the signs in the second term align and so it is simply a sum of PSD matrices. Hence, we conclude that $\bar{f}$ is convex over Ω as a function of $\mathbf{x}$.

Second, $\bar{f}$ coincides with f up to first order at $\mathbf{x} = \mathbf{x}_0$, as can be verified directly from the definitions.

Third, using the properties of majorizers (minorizers) and the correspondences between the entries of $[\mathbf{g}]$ and the monotonicity pattern of $\bar{h}$ we see that

$$\bar{f}(\mathbf{x};\mathbf{x}_0) = \bar{h}([\mathbf{g}](\mathbf{x};\mathbf{x}_0);\mathbf{u}_0) \geq \bar{h}(\mathbf{g}(\mathbf{x});\mathbf{u}_0) \geq h(\mathbf{g}(\mathbf{x}))$$

for all $\mathbf{x} \in \Omega(\mathbf{x}_0)$. Therefore, $\bar{f}$ is a convex majorizer of f at $\mathbf{x}_0$ over the set $\Omega(\mathbf{x}_0)$. □

This construction provides a non-trivial convex local majorizer of f. Then, in each step of Algorithm 1 we use this construction and set

$$H = \nabla^2_{\mathbf{x}} \bar{f}(\mathbf{x};\mathbf{x}_0)|_{\mathbf{x}=\mathbf{x}_0},$$

which boils down to equation (9).

3.3 Remarks

Structure of $\Omega(\mathbf{x}_0)$. In case all $s_j(\mathbf{u}_0) \neq 0$, the point $\mathbf{x}_0$ is in the strict interior of $\Omega(\mathbf{x}_0)$. In the non-generic case of equality in one or more of these equations, $\mathbf{u}_0$ is critical w.r.t. these coordinates (respectively) and $\mathbf{x}_0$ is at the boundary of $\Omega(\mathbf{x}_0)$. Nonetheless, it is interesting to note that the definition of the sign function we used, equation (13), is somewhat arbitrary for $t = 0$; we could alternatively set $\text{sign}(0) = -1$. In fact, we could use different definitions of the sign in each coordinate j of (14). Thus, the equations $\partial\bar{h}/\partial u_j = 0$ implicitly define a partition of the neighborhood of $\mathbf{x}_0$, similar to orthants around the origin, wherein f is majorized over each subset with a compatible choice of sign functions. This further implies that the Hessian at $\mathbf{x}_0$ is well defined, as these majorizers all coincide up to the second order at $\mathbf{x}_0$, see equation (17).

Hessian-only computation. We note that the convex-concave decompositions of h_i and $\mathbf{g}_i$ were necessary for the derivation, but are in fact not needed for forming a PSD Hessian using our formula in equation (9). Rather, this computation only requires convex-concave decompositions of the Hessians,

$$\nabla^2 h_i = \nabla^2 h_i^+ + \nabla^2 h_i^-; \quad \nabla^2 \mathbf{g}_i = \nabla^2 \mathbf{g}_i^+ + \nabla^2 \mathbf{g}_i^-,$$

where $\nabla^2 h_i^+, -\nabla^2 h_i^-, \nabla^2 \mathbf{g}_i^+$ and $-\nabla^2 \mathbf{g}_i^-$ are all PSD matrices. Note, however, that the claims of Proposition 3.1 are not guaranteed to hold in case of a convex-concave decomposition of Hessians that does not correspond to a global functional convex-concave decomposition.

Relation to generalized Gauss-Newton. It is interesting to note that the first term of equation (9) is exactly the generalized Gauss-Newton matrix [Schraudolph 2002], which involves second order derivatives of h_i but only first order derivatives of $\mathbf{g}_i$. From that aspect, our approach supplements the generalized Gauss-Newton approximate Hessian by adding positive semidefinite terms accounting for the second order derivatives of $\mathbf{g}_i$ as well.

Hessian approximation and term gathering. Proposition 3.1 implies that the derived PSD Hessian $H \succeq 0$ satisfies $H \succeq \nabla^2 f(\mathbf{x}_0)$ (inequalities are in the PSD sense). A natural question is whether equality holds when $\nabla^2 f(\mathbf{x}_0) \succeq 0$. Unfortunately, this is generally not the case, similarly to other non-full Hessian projection methods.

A tighter approximation to the Hessian can be achieved by term gathering, as we explain next. Note that the second and third terms of equation (9) are of the basic form

$$M = (a)_+ M_1 + (b)_+ M_2,$$

where M_1 and M_2 are both PSD. We observe that if M_1, M_2 share a common term P, such that $M_1 - P$, $M_2 - P$ and P are all PSD, then since $(a+b)_+ \leq (a)_+ + (b)_+$ we have

$$M \succeq (a)_+(M_1 - P) + (b)_+(M_2 - P) + (a+b)_+ P \succeq 0. \tag{18}$$

Thus, term gathering produces a Hessian approximation H' that satisfies $H \succeq H' \succeq 0$ and $H' \succeq \nabla^2 f(\mathbf{x}_0)$. Empirically, we observed that term gathering may improve convergence by a small margin.

4 ENERGIES ON TRIANGULAR MESHES

Variational approaches in geometry processing often aim to minimize energies related to some notion of distortion. They often strive to approximate an isometric map [Aigerman et al. 2015; Chao et al. 2010; Liu et al. 2008; Smith and Schaefer 2015; Sorkine and Alexa 2007], conformal map [Ben-Chen et al. 2008; Desbrun et al. 2002; Hormann and Greiner 2000; Lévy et al. 2002; Mullen et al. 2008; Weber et al. 2012], or a harmonic map [Ben-Chen et al. 2009]. Other approaches minimize energies that represent physical quantities [Wardetzky et al. 2007; Xu et al. 2015].

Due to their prevalence in geometry processing we concentrate on triangular meshes. We represent a piecewise linear mapping of a surface mesh with N vertices into the plane as a vector $\mathbf{x} \in \mathbb{R}^{2N}$, encoding the image of each of the vertices. We further denote by

$$J_i = J_i(\mathbf{x}) = \begin{bmatrix} a_i(\mathbf{x}) & b_i(\mathbf{x}) \\ c_i(\mathbf{x}) & d_i(\mathbf{x}) \end{bmatrix} \tag{19}$$

the 2×2 Jacobian matrix, with respect to an arbitrary local frame (*i.e.*, deformation gradient), associated with the i-th mesh triangle, and note that its entries a_i, b_i, c_i, d_i are linear in $\mathbf{x}$.

Singular value energy templates. Energies defined in terms of the singular values of J_i are prevalent in geometry processing. They

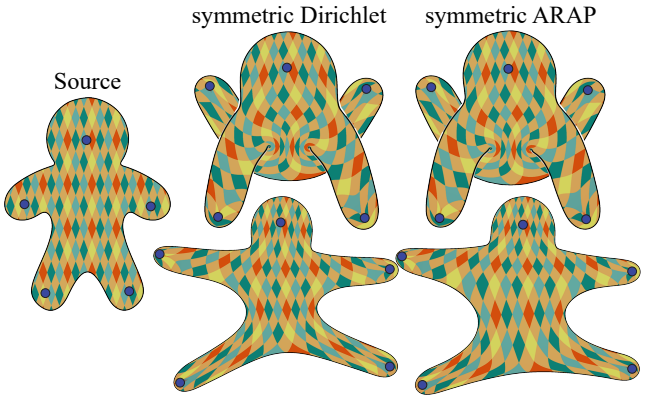


Fig. 3. Shape deformation with positional constraints, using different energies. The symmetric Dirichlet and symmetric ARAP energies are both minimized using our method, demonstrating how it can be used for minimizing different distortion energies.

can often be formulated as

$$f(\mathbf{x}) = \sum_i h(\Sigma_i, \sigma_i) |t_i|, \tag{20}$$

where summation is over all triangles, Σ_i and σ_i are the singular values of the Jacobian J_i of the i-th triangle, and $|t_i|$ is its area.

Energies of the type (20) are already in the form our approach addresses, namely, $\sum_i h_i \circ \mathbf{g}_i$ with

$$\mathbf{g}_i = (\Sigma_i, \sigma_i). \tag{21}$$

To obtain a convex-concave decomposition for $\mathbf{g}_i$ recall that [Lipman 2012; Smith and Schaefer 2015] use the fact that

$$\begin{aligned}\Sigma_i &= \|\alpha_i\| + \|\beta_i\| \\ \sigma_i &= \|\alpha_i\| - \|\beta_i\|,\end{aligned}$$

where,

$$\alpha_i = \frac{1}{2}\begin{bmatrix} a_i + d_i \\ c_i - b_i \end{bmatrix}, \quad \beta_i = \frac{1}{2}\begin{bmatrix} a_i - d_i \\ c_i + b_i \end{bmatrix} \tag{22}$$

represent the closest similarity (complex derivative) and closest anti-similarity (complex anti-derivative), respectively [Chien et al. 2016].

Therefore, it is only natural to decompose $\mathbf{g}_i = \mathbf{g}_i^+ + \mathbf{g}_i^-$ with

$$\begin{aligned}\mathbf{g}_i^+ &= (\|\alpha_i\| + \|\beta_i\|, \|\alpha_i\|) \\ \mathbf{g}_i^- &= (0, -\|\beta_i\|).\end{aligned} \tag{23}$$

Clearly, $\mathbf{g}_i^+$ is convex and $\mathbf{g}_i^-$ is concave. Note that $\mathbf{g}_i^+$ and $\mathbf{g}_i^-$ have $\|\alpha_i\|$ and $\|\beta_i\|$ in common; and thus, in our implementation, we use the common term gathering in equation (18).

Next, we apply this template for two energies that measure isometric distortion.

Symmetric Dirichlet energy. The isometric distortion energy considered in [Schreiner et al. 2004; Smith and Schaefer 2015] is

$$f_{\text{ISO}}(\mathbf{x}) = \sum_i \left(\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}\right) |t_i|. \tag{24}$$

It immediately takes the form $\sum_i h_i \circ \mathbf{g}_i$ with $\mathbf{g}_i = (\Sigma_i, \sigma_i)$ as in equation (21) and

$$h_i(u, v) = \left(u^2 + u^{-2} + v^2 + v^{-2}\right) |t_i|.$$

Note that h_i is convex with respect to $u, v > 0$ and thus, with the decomposition (23) of $\mathbf{g}_i$ the formula for H in equation (9) can be readily computed to obtain a PSD Hessian. Figure 3 depicts mesh deformations obtained by minimizing this energy.

Symmetric as-rigid-as-possible (ARAP) energy. This isometric distortion energy is a symmetric version of the ARAP energy that equally penalizes stretching and shrinking and hence also resists element flips. We formulate it as

$$f_{\text{SARAP}}(\mathbf{x}) = \sum_i \left((\Sigma_i - 1)^2 + \left(\sigma_i^{-1} - 1\right)^2\right) |t_i|. \tag{25}$$

It takes the form $\sum_i h_i \circ \mathbf{g}_i$ with $\mathbf{g}_i = (\Sigma_i, \sigma_i)$ and

$$h_i(u, v) = \left((u - 1)^2 + \left(v^{-1} - 1\right)^2\right) |t_i|.$$

Note that while the term dependent on u in h_i is convex, the term dependent on v is not. Since $(v^{-1} - 1)^2$ is convex on $(0, 1.5]$ and concave on $(1.5, \infty)$, it is easy to analytically decompose it, using its linearization at $v = 1.5$, as (omitting the area term for brevity)

$$h^+(u, v) = \begin{cases} (u-1)^2 + \left(v^{-1} - 1\right)^2 & 0 < v \le 1.5 \\ (u-1)^2 - \frac{3}{9} + \frac{8}{27}v & v > 1.5 \end{cases}$$

$$h^-(u, v) = \begin{cases} 0 & 0 < v \le 1.5 \\ \left(v^{-1} - 1\right)^2 + \frac{3}{9} - \frac{8}{27}v & v > 1.5 \end{cases}$$

In turn, the formula for H in equation (9) can be readily used for minimizing the symmetric ARAP energy, see Figure 3 for an example.

Strain energy density for Neo-Hookean material. In two dimensions, the strain energy density for Neo-Hookean material can be formulated as

$$f_{\text{NH}}(\mathbf{x}) = \sum_i \left[\frac{\mu}{2}\left(\frac{\|J_i\|_F^2}{\det J_i} - 2\right) + \frac{\kappa}{2}(\det J_i - 1)^2\right] |t_i|, \tag{26}$$

where μ and κ are modeling coefficients, related to the response of the material to shear stress and compression resistance, respectively [Xu et al. 2015]. Geometrically, the first term of the objective is related to conformal distortion (MIPS) [Hormann and Greiner 2000], whereas the second term accounts for area change.

This energy takes the form $\sum_i h_i \circ \mathbf{g}_i$ with

$$\begin{aligned} h_i(u, v) &= \left(\frac{\mu}{2}\frac{u^2}{v} + \frac{\kappa}{2}(v - 1)^2\right) |t_i| \\ \mathbf{g}_i(\mathbf{x}) &= (\|J_i\|_F, \det J_i). \end{aligned}$$

Note that h_i is convex and so is g_{i1}. For its second entry we note that, with the notations of equation (19), we have

$$\begin{aligned} &\det J_i = a_i d_i - b_i c_i \\ &= \frac{1}{4}\left[(a_i + d_i)^2 + (b_i - c_i)^2\right] - \frac{1}{4}\left[(a_i - d_i)^2 + (b_i + c_i)^2\right]. \end{aligned}$$

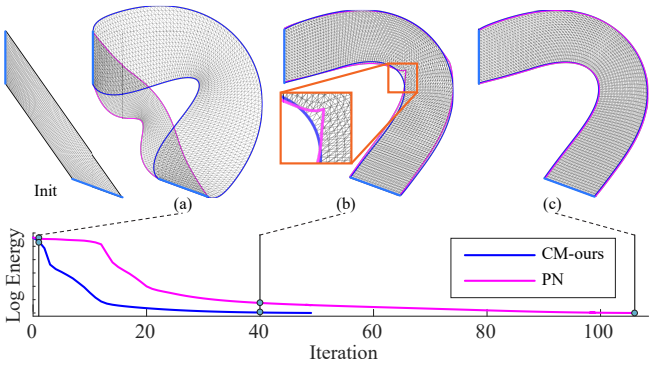


Fig. 4. Minimization of the Neo-Hookean strain density using our approach. Positional constraints are depicted in light blue. Snapshots of the state of the mesh are presented for our method (blue) and projected Newton (PN) (magenta) after (a) 1 iteration, (b) 40 iterations, and (c) upon the convergence of PN. Note that our approach is substantially more effective in minimizing this energy from the initial state (left, in black).

Thus we take

$$\mathbf{g}_i^+(\mathbf{x}) = \left(\|J_i\|_F \,,\ \frac{1}{4}\left[(a_i+d_i)^2 + (b_i-c_i)^2\right] \right)$$
$$\mathbf{g}_i^-(\mathbf{x}) = \left(0,\ -\frac{1}{4}\left[(a_i-d_i)^2 + (b_i+c_i)^2\right] \right).$$

The advantage of using our approach for this energy is demonstrated in Figure 4; see Section 5.2 for additional details.

5 EXPERIMENTAL EVALUATION

We tested our approach on a number of standard problems in geometry processing formulated using the energies introduced in the previous section. We compared our approached (**CM-ours**) to the following state of the art approaches:

Newton's Algorithm – We followed the approach described in [Liu et al. 2016; Teran et al. 2005] and implemented a projected Newton (**PN**) solver, wherein the Hessian of each triangle is computed using automatic differentiation, and then individually projected via eigen-decomposition onto the PSD cone. We used our analytic Hessian formula in equation (9) to efficiently compute the indefinite Hessian (by simply dropping the clamping).

We observed that computation of Hessians using our analytic formulation works an order of magnitude faster than the automatic differentiation typically used for this task [Liu et al. 2016; Rabinovich et al. 2017].

Scalable Locally Injective Mappings – We implemented the approach of [Rabinovich et al. 2017], wherein a modified Laplacian takes the place of the Hessian (**SLIM**). We followed the authors' update formula for the reweighting.

Accelerated Quadratic Proxy – We implemented the approach of [Kovalsky et al. 2016], wherein the Laplacian itself replaces the Hessian. We compared to their accelerated (**AQP**) as well as non-accelerated (**QP**) variants.

Implementation details. For comparability, we integrated the above approaches into a single C++ implementation of Algorithm 1, deployed on an Intel i7-3970X, CPU 3.50GHz machine. Our implementation relies on LIBIGL [Jacobson et al. 2016]; we employ the PARDISO solver [Kuzmin et al. 2013; Schenk et al. 2008, 2007] for the linear solve in the CM-ours, PN and SLIM algorithms, and use LU prefactorization for the QP and AQP algorithms.

We used a standard Armijo backtracking algorithm for determining the step size [Nocedal and Wright 2006]. The energies we used infinitely penalize for triangle inversion, see Section 4. As such, we used the determinant criterion of [Smith and Schaefer 2015] to determine a maximal non-inverting step size. This ensures that, in each of the iterations of Algorithm 1, the initial triangle orientations are all preserved.

5.1 Surface parameterization

We tested our approach on surface parameterization computed by minimizing the symmetric Dirichlet energy, equation (24). We followed the standard protocol of using Tutte's embedding for computing a bijective initial parameterization [Kovalsky et al. 2016; Rabinovich et al. 2017; Smith and Schaefer 2015]. In turn, we note that the resulting parameterizations are guaranteed to be locally injective, as we minimize an inversion-resisting energy and accordingly restrict our line-search.

We conducted an extensive evaluation of our approach on a dataset of 30 surfaces taken from [Myles et al. 2014; Rabinovich et al. 2017]. Figure 5 shows the results obtained in four examples, comparing the performance of our approach with that of alternative approaches. Our approach (blue) outperforms the others by a significant margin. Table 1 further summarizes our evaluation. Noticeably, the second order approaches (CM-ours, PN) require substantially fewer iterations to converge. Consequently, their overall convergence time is also lower, even though the first order alternatives (SLIM, AQP) spend less time per iteration.

Figure 6 exemplifies the scalability of our approach in comparison to that of SLIM. In this experiment, we computed the parameterization of meshes of different sizes obtained by coarsening a high resolution surface mesh. Our method converges after an almost constant number of iterations, a behavior that is typical to second order optimization approaches, and scales substantially better than SLIM to high resolution meshes.

5.2 Mesh deformation

We evaluated our approach for computing mesh deformations. Similarly to surface parameterization, our method demonstrates superior performance. Figure 1 shows a comparative experiment where we minimize the symmetric ARAP of equation (25): we first bend a bar, then release it and measure the time it takes to reach the rest pose. In this example our method is more than 4 times faster than projected Newton and substantially faster than the others. Figure 3 demonstrates the deformations obtained by minimizing the symmetric Dirichlet and symmetric ARAP energies using our method.

We further argue that, for deformation, short per iteration time is as important as convergence rate. Since deformation is an interactive process, it must be reactive to the input of the user. In our experiment we observed that our iterations are also faster than the projected

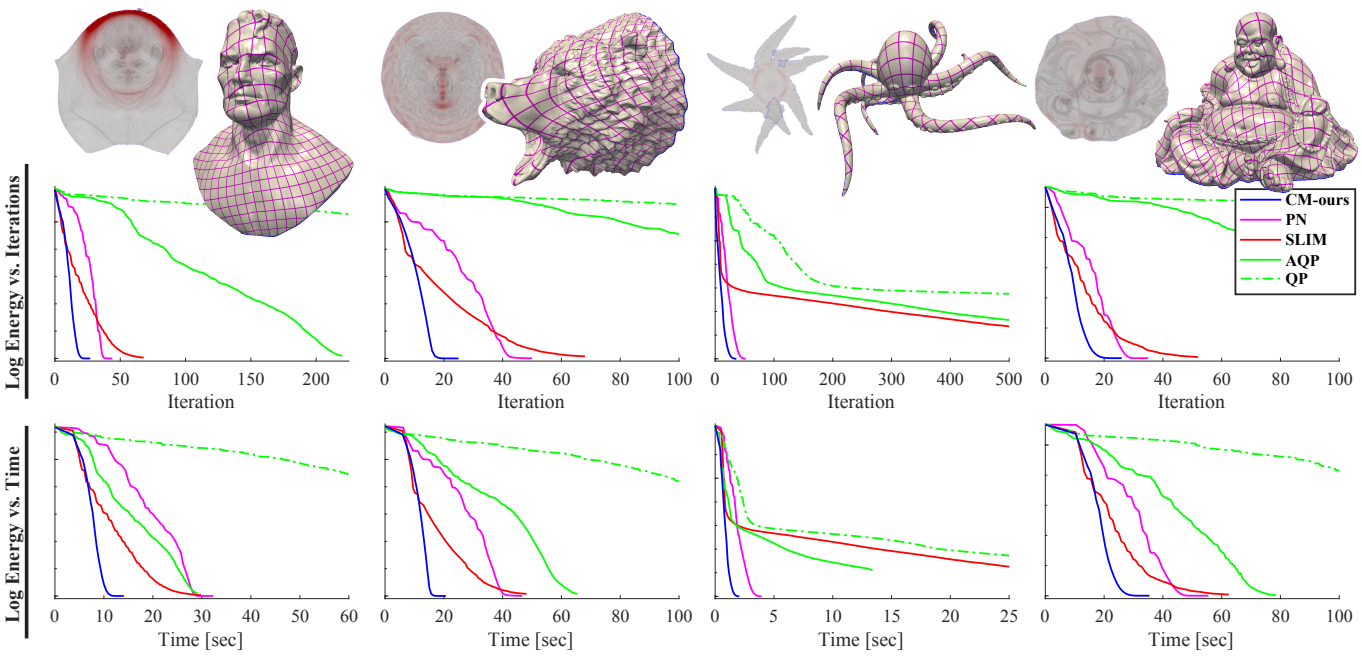


Fig. 5. Surface parameterization: comparing our approach with the state of the art for the minimization of the symmetric Dirichlet Energy. The graphs show how the different optimization approaches reduce the energy as a function of the number of iterations or run time. Noticeably, our approach (blue) outperforms the alternative approaches. The top row shows the planar parameterization obtained using our optimization (shading indicating isometric distortion) and a texture respectively mapped back to the surface.

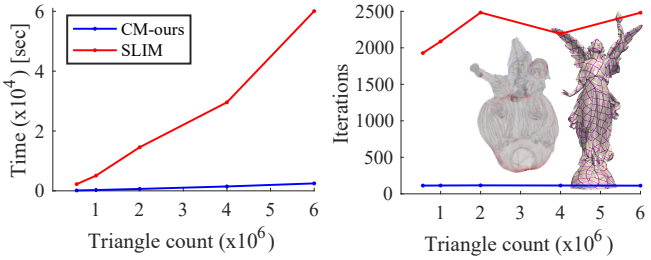


Fig. 6. Scalability. We applied our approach and SLIM for parameterizing meshes of increasing resolutions representing the same surface. The graphs show the number of iterations and run time required for problems of different sizes to converge.

Newton iterations and thus better suited for deformation; this is exemplified in the attached supplemental video.

Figure 4 demonstrates how our approach is used for minimizing the Neo-Hookean strain density in equation (26). In this experiment we set the energy parameters to $\mu/\kappa = 2 \cdot 10^{-3}$, which models natural rubber. We initialize with a feasible solution that satisfies the positional constraints illustrated in the figure. The projected Newton algorithm requires more than twice as many iterations to converge. In addition, our approach is more effective in reducing the energy in the first few iterations; the figure shows the result after just 1 iteration, 40 iterations and upon the convergence of the projected Newton approach.

5.3 Seamless parameterization

Introducing compatibility constraints on the seams of a cut surface finds applications in quadrangulation [Bommes et al. 2009] and surface mapping [Aigerman et al. 2015] among others. We have experimented with supplementing our surface parameterization algorithm (Section 5.1) with two types of seam compatibility constraints:

Similarity – where we constrain corresponding edges that belong to the same seam, to be related by a similarity transformation. Namely, if $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^2$ are the end-points of a seam edge corresponding to $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^2$, we require that

$$\mathbf{v}_2 - \mathbf{v}_1 = T(\mathbf{u}_2 - \mathbf{u}_1) = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} (\mathbf{u}_2 - \mathbf{u}_1), \tag{27}$$

where T is a *similarity* matrix parameterized by $a, b \in \mathbb{R}$.

Seamless – where the matrix T above is constrained to be a *rotation* by a multiple of 90 degrees. We realize this constraint by enforcing equation (27) with the *additional* quadratic constraints

$$a^2 + b^2 = 1; \quad ab = 0. \tag{28}$$

Note that this compatibility constraint is a fundamental prerequisite for producing a quadrangulation, however, it is insufficient on its own [Bommes et al. 2013].

Figure 7 illustrates the difference between parameterizations that satisfy either of these compatibility constraints.

			CM-ours			Projected Newton			SLIM			AQP		
Name	# vert	# elem	iter	time [sec]	energy	iter	time [sec]	energy	iter	time [sec]	energy	iter	time [sec]	energy
cow	4540	8626	33	**0.60**	2.08413	41	0.98	2.08413	394	5.63	2.08417	316	1.63	2.08420
bumpy_sphere	5851	11444	20	0.51	2.04661	19	0.68	2.04661	122	2.26	2.04662	61	**0.41**	2.04662
dente	9512	18670	56	2.01	2.02605	37	**1.94**	2.02605	384	12.94	2.02613	-	-	-
octo2	15141	26968	36	**2.00**	2.09024	53	3.88	2.09024	812	37.52	2.09027	800	13.37	2.09036
hilbert	18424	32228	80	**4.24**	2.00001	112	8.70	2.00000	1466	67.58	2.00050	613	10.47	2.00023
armadillo1	22145	43160	29	**3.07**	2.07490	44	5.73	2.07490	339	28.21	2.07494	-	-	2.07491
david	25604	50864	22	**2.68**	2.19244	27	4.14	2.19244	107	10.87	2.19245	312	8.57	2.19246
blade	29583	58546	27	**4.10**	2.08289	28	5.42	2.08289	327	40.82	2.08291	355	12.98	2.08293
hand	37234	72958	23	**4.21**	2.01556	27	6.19	2.01556	391	55.44	2.01557	399	17.83	2.01557
gargoyle	49622	98803	34	**8.47**	4.21048	47	14.61	4.21048	190	40.50	4.21049	-	-	4.21048
vase_lion	49862	99237	27	**6.15**	3.51237	39	11.07	3.51237	104	20.18	3.51237	-	-	3.51237
armchair	50399	100000	21	**5.07**	2.07815	25	7.65	2.07815	360	67.30	2.07816	407	24.00	2.07824
bimba	50296	100000	24	**6.29**	2.34664	30	9.67	2.34664	223	46.70	2.34665	319	20.42	2.34665
buste	50390	100000	38	**8.13**	2.13799	35	10.19	2.13799	533	97.33	2.13800	1041	62.92	2.13799
camille_hand	50608	100000	37	**7.93**	2.05489	48	13.46	2.05489	872	161.59	2.05506	3886	255.80	2.05546
chinese_lion	50365	100000	40	**9.56**	2.23405	44	13.45	2.23405	459	96.40	2.23407	1199	85.15	2.23424
armadillo2	50401	100000	63	**13.15**	2.48220	75	21.04	2.48220	1022	199.90	2.48226	-	-	2.48221
bunny1	54066	107936	108	**28.80**	4.22693	110	37.81	4.22693	660	172.05	4.22698	2219	163.30	4.22709
bunny2	56250	111364	32	**9.53**	2.05018	57	20.23	2.05018	678	173.04	2.05021	1906	174.01	2.05031
superman2	95648	190471	29	**14.54**	3.16845	43	25.69	3.16845	71	31.21	3.16845	478	62.89	3.16846
superman3	95712	190471	27	**13.66**	2.61709	56	31.84	2.61709	68	29.42	2.61709	220	28.98	2.61709
superman1	101665	203002	32	**16.81**	3.98495	59	37.18	3.98495	105	48.14	3.98496	-	-	3.98495
male	148031	293224	87	**57.03**	2.04748	101	84.27	2.04748	1039	656.08	2.04757	4116	970.33	2.04879
bear	148484	296409	25	**20.07**	2.74587	49	45.78	2.74587	68	47.49	2.74587	267	65.06	2.74587
octo1	151415	299324	40	**25.71**	2.08371	48	39.66	2.08371	979	536.40	2.08379	1554	332.23	2.08394
dragon_head	194093	386992	21	**23.39**	2.38146	25	32.25	2.38146	71	63.58	2.38147	508	161.62	2.38147
eros	197405	394456	26	**31.43**	4.65212	27	39.76	4.65212	148	157.73	4.65214	1155	348.42	4.65214
buddha	235771	470507	26	**34.39**	2.36141	35	54.18	2.36141	52	61.38	2.36141	186	77.84	2.36141
lucy	501130	1000000	117	**293.93**	2.23208	153	455.03	2.23208	2579	6365.07	2.23235	-	-	2.24268
chinese_dragon	657331	1311956	40	**129.96**	2.25100	42	164.68	2.25100	1148	3176.33	2.25107	-	-	4.61829

Table 1. Surface parameterization: comparing our approach with state-of-the-art approaches for the minimization of the symmetric Dirichlet Energy. Experiments that did not converge within 6000 iterations are omitted from the table. Our algorithm compares favorably in both iteration count and convergence time in all cases excluding two instances where it is found to be comparable. Note that SLIM and AQP typically fail to reduce the energy to the values achieved by our approach.

We have computed parameterizations with seam compatibility constraints on surfaces from the dataset of [Myles et al. 2014]. We cut each of the surfaces along geodesics corresponding to a minimal spanning tree defined by the singularity points provided with the dataset. We then used our Algorithm 1 with our choice of Hessian to minimize the symmetric Dirichlet energy.

In our first experiment we computed maps subject to similarity seam compatibility constraints. To that end, we added a term to the objective function that penalizes, in the least squares sense, for the bilinear constraints of equation (27); respectively, we added the Gauss-Newton matrix [Nocedal and Wright 2006] corresponding to this penalty term to the Hessian we used. We empirically observed that adding this term with a small fixed weight (0.1 in our experiments) suffices to satisfy the similarity seam compatibility constraints (up to numerical precision); results are shown in Figure 7 (left) and Figure 8 (top).

Our second experiment aimed to enforce a seamless compatibility constraint, *i.e.*, restrict the transformation relating corresponding edges of a seam to a rotation by a multiple of 90 degrees. We achieved that by introducing an additional objective term penalizing for the quadratic constraints of equation (28). We started from the parameterization satisfying the similarity seam compatibility constraints as described above. Then, we solved a sequence of optimization problems, wherein we gradually increased the weights of the terms penalizing for the seam compatibility constraints in equations (27)-(28). Our experiments used the same naive penalty schedule for all examples (weights increased from 10^{-5} by a factor of 10 until all constraints are met); also, we did not utilize any additional information (*e.g.*, prescribed rotations extracted from a vector field, as in [Rabinovich et al. 2017]). Figure 7 (right) and Figure 8 (bottom) show results obtained using our approach.

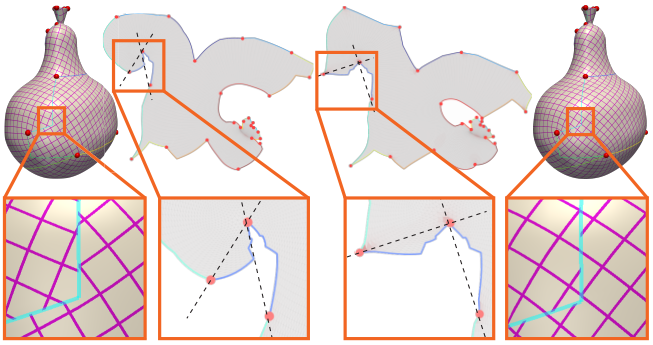


Fig. 7. Parameterization with seam compatibility constraints. We enforce either a similarity transformation between corresponding seam edges (left) or a rotation by a multiple of 90 degrees (right). The illustrations on the flat parameterization depict corresponding edges of a seam related by either a similarity or a rotation by 90 degrees. Note that the latter suffices to ensure that angles are correctly aligned, but still misses integer translations required for quadrangulation [Bommes et al. 2013].

6 CONCLUDING REMARKS

In this paper we presented an optimization approach for energies that are compositions of simpler parts, each admitting a convex-concave decomposition. Our approach exploits this structure to devise a novel convex majorizer, and consequently an efficient second order optimization algorithm. Thus, each iteration of our algorithm is in a sense aware of the global energy landscape, rather than relying on strictly local second order approximation. We demonstrated the utility of our approach in surface parameterization and mesh deformation, where its performance exceeds the state of the art.

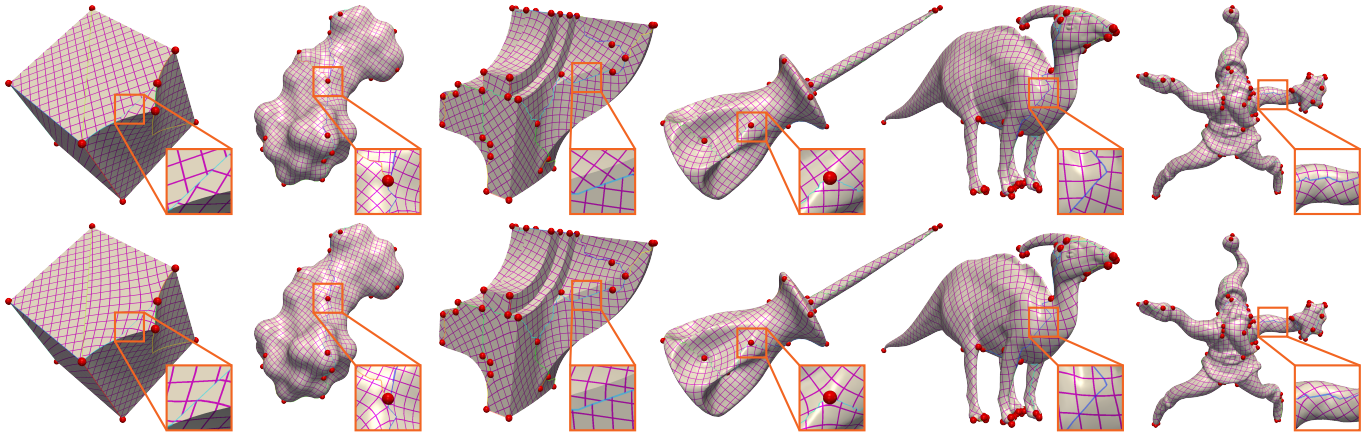


Fig. 8. Parameterization with seam compatibility constraints. We enforce either a similarity transformation between corresponding seam edges (top) or a rotation by a multiple of 90 degrees (bottom). Note that the latter suffices to ensure that angles are correctly aligned, but is insufficient on its own to produce an integer grid map inducing quadrangulation [Bommes et al. 2013].

One limitation of our approach is that the majorizer we define depends on the particular decomposition used, *i.e.*, different decompositions of the objective function generally correspond to different majorizers, and thus to different choices of H. In this aspect, we note that the notions of optimality or "quality" of H are ill-defined; we are unaware of a method for comparing different choices of H that predicts their effectiveness for optimization. For instance, we observed that the difference between our choice of H and that of projected Newton's is an indefinite matrix.

We also note that our choice of Hessian in equation (9) is, in general, not guaranteed to coincide with the true Hessian of the objective function on its convex regions. In some cases, this can be directly revealed by the functions composing our majorizer. Detecting this discrepancy can perhaps be used, in future work, to improve convergence, *e.g.*, by reverting to the true Hessian whenever possible.

Lastly, the approach presented in this work can be extended and applied to problems in higher dimension. In these cases, we believe that our analytic formula would provide an even better relative performance improvement over approaches such as projected Newton, that need to project (*e.g.*, via eigen-decomposition) larger matrices.

ACKNOWLEDGMENTS

This research was supported in part by the European Research Council starting grants Surf-Comp (Grant No. 307754) and iModel (Grant No. 306877), I-CORE program of the Israel PBC and ISF (Grant No. 4/11) and the Simons Foundation Math+X Investigator award. The authors would like to thank Olga Diamanti, Michael Rabinovich and Ashish Myles for sharing their code; Amir Porat for producing the supplemental video; and the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

Noam Aigerman, Roi Poranne, and Yaron Lipman. 2015. Seamless surface mappings. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 72.

Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 449–458.

Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. 2009. Variational harmonic maps for space deformation. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 34.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013. Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 98.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. In *ACM Transactions On Graphics (TOG)*, Vol. 28. ACM, 77.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 38.

Edward Chien, Renjie Chen, and Ofir Weber. 2016. Bounded distortion harmonic shape interpolation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 105.

Ingrid Daubechies, Michel Defrise, and Christine De Mol. 2004. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics* 57, 11 (2004), 1413–1457.

Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinan Güntürk. 2010. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics* 63, 1 (2010), 1–38.

Mathieu Desbrun, Mark Meyer, and Pierre Alliez. 2002. Intrinsic parameterizations of surface meshes. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 209–218.

Xiao-Ming Fu and Yang Liu. 2016. Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 216.

Emden R Gansner, Yehuda Koren, and Stephen North. 2004. Graph drawing by stress majorization. In *International Symposium on Graph Drawing*. Springer, 239–250.

Philip Hartman and others. 1959. On functions representable as a difference of convex functions. *Pacific J. Math* 9, 3 (1959), 707–713.

Kai Hormann and Günther Greiner. 2000. *MIPS: An efficient global parametrization method*. Technical Report. DTIC Document.

Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. 2009. Shape decomposition using modal analysis. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 407–416.

David R Hunter and Kenneth Lange. 2004. A tutorial on MM algorithms. *The American Statistician* 58, 1 (2004), 30–37.

Alec Jacobson, Daniele Panozzo, and others. 2016. LibIGL: A simple C++ geometry processing library. (2016). http://libigl.github.io/libigl/.

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

A. Kuzmin, M. Luisier, and O. Schenk. 2013. Fast Methods for Computing Selected Elements of the Green's Function in Massively Parallel Nanoelectronic Device Simulations. In *Proc. Euro-Par*. 533–544.

Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (ToG)* 24, 3 (2005), 795–802.

K. Lange. 2013. *Optimization*. Springer.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 362–371.

Yaron Lipman. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 108.

Yaron Lipman, Stav Yagev, Roi Poranne, David W Jacobs, and Ronen Basri. 2014. Feature matching with bounded distortion. *ACM Transactions on Graphics (TOG)* 33, 3 (2014), 26.

Thomas Lipp and Stephen Boyd. 2014. Variations and extension of the convex–concave procedure. *Optimization and Engineering* (2014), 1–25.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. 2008. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1495–1504.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2016. Towards Real-time Simulation of Hyperelastic Materials. *arXiv preprint arXiv:1604.07378* (2016).

James Martens. 2010. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 735–742.

Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2008. Spectral conformal parameterization. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1487–1494.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-aligned Global Parametrization. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), Article No. 135.

Jorge Nocedal and Stephen Wright. 2006. *Numerical Optimization*. Springer Science & Business Media.

Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics* 2, 1 (1993), 15–36.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Maps. *ACM Transactions on Graphics (TOG)* 36, 2 (2017), 16:1–16:16.

Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. 2008. On Large-Scale Diagonalization Techniques for the Anderson Model of Localization. *SIAM Rev.* 50, 1 (2008), 91–112.

Olaf Schenk, Andreas Wächter, and Michael Hagemann. 2007. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Computational Optimization and Applications* 36, 2-3 (2007), 321–341.

Nicol N Schraudolph. 2002. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation* 14, 7 (2002), 1723–1738.

John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. 2004. Inter-surface mapping. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 870–877.

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

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 66.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4.

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with polyhedral meshes made simple. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 70–1.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 181–190.

Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. 2007. Discrete quadratic curvature energies. *Computer Aided Geometric Design* 24, 8 (2007), 499–518.

Ofir Weber, Ashish Myles, and Denis Zorin. 2012. Computing extremal quasiconformal maps. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1679–1689.

Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbic. 2015. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 75.

Alan L Yuille and Anand Rangarajan. 2003. The concave-convex procedure. *Neural Computation* 15, 4 (2003), 915–936.