# Large-Scale Bounded Distortion Mappings

Shahar Z. Kovalsky Noam Aigerman Ronen Basri Yaron Lipman
Weizmann Institute of Science

## Abstract

We propose an efficient algorithm for computing large-scale bounded distortion maps of triangular and tetrahedral meshes. Specifically, given an initial map, we compute a similar map whose differentials are orientation preserving and have bounded condition number.

Inspired by alternating optimization and Gauss-Newton approaches, we devise a first order method which combines the advantages of both. On the one hand, its iterations are as computationally efficient as those of alternating optimization. On the other hand, it enjoys preferable convergence properties, associated with Gauss-Newton like approaches.

We demonstrate the utility of the proposed approach in efficiently solving geometry processing problems, focusing on challenging large-scale problems.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry G.1.6 [Numerical Analysis]: Optimization;

**Keywords:** optimization, first order methods, bounded conformal distortion, bounded distortion mappings, simplicial meshes

## 1 Introduction

Computing simplicial mappings of large-scale meshes with unflipped and bounded aspect-ratio elements is of great interest in computer graphics and geometry processing. It is often required for applications such as surface and volume parameterization, remeshing and quadrangulation, and shape mapping.

The goal of this paper is to devise an *efficient* and *highly scalable* algorithm for computing bounded distortion maps: Given an initial simplicial map of a triangular or a tetrahedral mesh, it computes a similar map whose differentials are orientation preserving and have bounded condition number. In a sense, bounded distortion mappings naturally characterize well-behaved mappings. As such, an efficient and scalable approach for their computation is paramount for demanding geometry processing tasks, involving large-scale meshes or requiring interactive rates for moderate-scale problems, *e.g.*, a mesh containing over a million tetrahedra as in Figure 1.

This problem falls into the class of non-linear constrained optimization, which has been addressed using various techniques in geometry processing problems [Liu et al. 2008; Bouaziz et al. 2012; Aigerman and Lipman 2013; Schüller et al. 2013; Kovalsky et al. 2014; Tang et al. 2014; Balzer and Soatto 2014]. These techniques

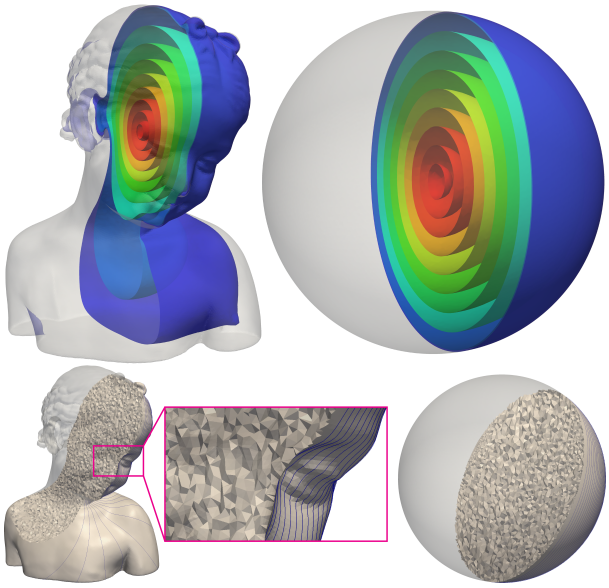


**Figure 1:** *Bijective volumetric mapping of mesh comprising 1.2M tetrahedra (tets). Initial mapping of the bust into a ball is obtained by minimizing the Dirichlet energy subject to a prescribed boundary map, resulting in over* $10\%$ *near-degenerate or flipped tets. A bounded distortion map is obtained using the proposed algorithm in 5 minutes. The mapping is depicted via corresponding isocontours (top) and a corresponding section through the volume (bottom).*

can be coarsely classified into first and second order methods. The latter, in particular interior point solver and other Newton variants, have been successfully employed for geometry processing problem (*e.g.*, [Aigerman and Lipman 2013; Schüller et al. 2013]. Although they benefit from the high order approximation, they typically scale poorly with problem size and fail to run on large-scale problems. First order methods mitigate this computational bottleneck by avoiding higher order information. Generally, they trade-off the preferable convergence properties of second order methods for significantly lower computational complexity.

Methods based on alternating projection and non-linear least squares are good representatives of first order techniques [Bouaziz et al. 2012; Tang et al. 2014; Balzer and Soatto 2014]. Alternating optimization techniques are considered to have the most computationally efficient iterations, often boiling down to the solution of a sparse linear system with a *constant* matrix. Their main drawback, however, is poor convergence properties; they often require a very large number of iterations in order to convergence. Non-linear least-squares is often solved with first-order Gauss-Newton or Levenberg-Marquardt approaches that offer better convergence properties via linearization. However, they require solving a *different* linear system in each iteration, which may become computationally prohibitive for large-scale problems. Moreover, they introduce parameters balancing the linearized terms, which may be non-trivial to properly set.

In this paper we develop a first-order method for solving the problem described above. It is inspired by alternating optimization and Gauss-Newton approaches and, in a sense, combines the advan-

tages of both. On the one hand, its iterations are as computationally efficient as those of alternating optimization. On the other hand, it enjoys preferable convergence properties, comparable with those of Gauss-Newton like approaches. Moreover, our method is parameter free and need not be tuned. While the proposed algorithm is not guaranteed to find a solution, in practice it performs well; its utility and robustness are demonstrated in evaluations and geometry processing applications, focusing on challenging large-scale problems.

## 2 Preliminaries and notations

In this work we consider mappings of triangular and tetrahederal meshes. More generally, consider a $d$-dimensional simplical complex with $n$ vertices $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ and $m$ simplices. That is, a triangular mesh for $d = 2$ and tetrahedral mesh for $d = 3$. Mapping the vertices to new positions, $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{d \times n}$, defines a piecewise linear (simplicial) mapping of the complex into $\mathbb{R}^d$, whereby the $j$-th simplex undergoes an affine map $\phi_j(\mathbf{v}) = C_j\mathbf{v} + \boldsymbol{\delta}_j$, where $C_j \in \mathbb{R}^{d \times d}$ and $\boldsymbol{\delta}_j \in \mathbb{R}^{d \times 1}$.

We adopt a column-stack notation $\mathrm{vec}\,(\cdot)$ and let $\mathbf{x} = \mathrm{vec}\,(\mathbf{U}) \in \mathbb{R}^{nd \times 1}$ represent the mapping of $\mathbf{V}$ to $\mathbf{U}$. We note that the matrices $C_1, \ldots, C_m$ can be expressed linearly in terms of $\mathbf{U}$ (*e.g.*, [Kovalsky et al. 2014]). We therefore set $T_j \in \mathbb{R}^{d^2 \times nd}$ to be the sparse matrix that maps $\mathbf{x}$ to $\mathrm{vec}\,(C_j)$, namely, $T_j\mathbf{x} = \mathrm{vec}\,(C_j)$. $T_j$ can be interpreted as the (linear) discrete differential operator which maps target coordinates of vertices to the differentials of the piecewise linear map they induce on the $j$-th simplex.

Lastly, we set $T \in \mathbb{R}^{md^2 \times nd}$ to be the vertical concatenation of $T_1, \ldots, T_m$, namely $T = [T_1\,;\, T_2\,;\, \cdots\,;\, T_m]$. We say the operator $T$ *lifts* the variable $\mathbf{x}$ into a higher dimensional space of differentials. We let $\mathbf{z} = T\mathbf{x}$ denote the *lifted variable*.

## 3 Problem statement and approach

**Goal.** Given an input map $\mathbf{x}_0$, we aim to find a similar map $\mathbf{x}$ whose differentials $T\mathbf{x}$ satisfy prescribed constraints, and its vertex images $\mathbf{x}$ satisfy linear equality constraints. We formulate this as the following optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|T\mathbf{x} - T\mathbf{x}_0\|_2 && \text{(1a)} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} && \text{(1b)} \\ & T\mathbf{x} \in \mathcal{D} && \text{(1c)} \end{aligned}$$

where (1c) is shorthand for $T_j\mathbf{x} \in \mathcal{D}_j$, for $j = 1, \ldots, m$, and $\mathcal{D}_j$ represents constraints on the $j$-th differential $C_j$ of the map $\mathbf{x}$.

In this work we take $\mathcal{D}_j = \mathcal{D}^K$ to be the subset of *$K$-bounded distortion (BD)* $d \times d$ matrices. Namely, matrices with non-negative determinant and condition number (*i.e.*, ratio of maximal to minimal singular values) at most $K$. Intuitively, this characterizes, in a scale invariant manner, mappings that are locally non-degenerate and orientation preserving. $K$ is typically set in the range $(1, 10^4]$.

Generally, the problem of minimizing an energy subject to bounded distortion constraints is known to be difficult and computationally demanding. We focus on the problem of efficiently finding a bounded distortion approximation of an arbitrary pre-computed mapping $\mathbf{x}_0$. Often, such two-step approach provides a good approximation to direct bounded distortion optimization (see [Kovalsky et al. 2014] for their comparison with [Aigerman and Lipman 2013]). Moreover, the efficient computation of such an approximation is important for cases where direct bounded distortion optimization is computationally prohibitive or for algorithms that require a feasible initialization.

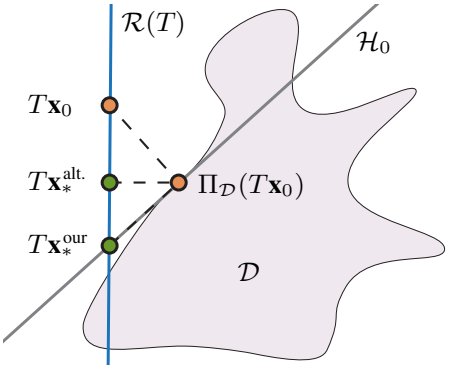



**Figure 2:** *Illustration of a single iteration in the lifted space of differentials. The blue line represents the range of the lift, $\mathcal{R}(T)$ – the subspace of feasible differentials; that is, differentials that can be realized by some choice of vertex positions. Current estimate $T\mathbf{x}_0$ is projected onto $\mathcal{D}$. An alternating optimization step $T\mathbf{x}_*^{\text{alt.}}$ finds the closest point on $\mathcal{R}(T)$. A step of our approach $T\mathbf{x}_*^{\text{our}}$ restricts the search to $\mathcal{H}_0$, attaining a closer point to $\mathcal{D}$.*

**Approach.** We devise a first-order iterative algorithm for approximating the solution of (1) inspired by the alternating and Gauss-Newton algorithms. It aims at superior convergence properties compared to the alternating approach, but at a similar computational cost.

The key idea is simple: We consider the set of non-linear constraints (1c) as a *single* high dimensional set $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_m$. Then we replace the inclusion $T\mathbf{x} \in \mathcal{D}$ with a linear proxy for $\mathcal{D}$; namely, we require that $T\mathbf{x}$ belongs to a *single hyperplane* $\mathcal{H}_0$ locally supporting $\mathcal{D}$ at the projection of $T\mathbf{x}_0$ onto $\mathcal{D}$ (see Figure 2). This results with the linearly constrained least squares,

$$\begin{aligned} \mathbf{x}_* = \underset{\mathbf{x}}{\text{argmin}} \quad & \|T\mathbf{x} - T\mathbf{x}_0\|_2 && \text{(2a)} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} && \text{(2b)} \\ & T\mathbf{x} \in \mathcal{H}_0 && \text{(2c)} \end{aligned}$$

We then set $\mathbf{x}_0 = \mathbf{x}_*$ and iterate until convergence.

The hyperplane $\mathcal{H}_0$ is uniquely defined by the Euclidean projection of $T\mathbf{x}_0$ onto $\mathcal{D}$, denoted by $\Pi_{\mathcal{D}}(T\mathbf{x}_0)$. It is taken to pass through $\Pi_{\mathcal{D}}(T\mathbf{x}_0)$ and be orthogonal to the projection direction $T\mathbf{x}_0 - \Pi_{\mathcal{D}}(T\mathbf{x}_0)$. Intuitively, $\mathcal{H}_0$ plays the role of a supporting hyperplane of the set $\mathcal{D}$ at the point $\Pi_{\mathcal{D}}(T\mathbf{x}_0)$. This construction and the resulting procedure are illustrated in Figure 2 and described in detail in Section 4.

**Relation to first-order approaches.** Replacing (1c) with (2c) reveals two key advantages of the algorithm. First, as with Gauss-Newton methods, it provides a higher approximation order for the constraint $T\mathbf{x} \in \mathcal{D}$. The Pythagorean theorem asserts that if $T\mathbf{x} \in \mathcal{H}_0$ then

$$\|T\mathbf{x} - T\mathbf{x}_0\|_2^2 = \|T\mathbf{x} - \Pi_{\mathcal{D}}(T\mathbf{x}_0)\|_2^2 + \|\Pi_{\mathcal{D}}(T\mathbf{x}_0) - T\mathbf{x}_0\|_2^2 .$$

As the second term is constant, problem (2) can be equivalently reformulated as

$$\begin{aligned} \mathbf{x}_* = \underset{\mathbf{x}}{\text{argmin}} \quad & \|T\mathbf{x} - \Pi_{\mathcal{D}}(T\mathbf{x}_0)\|_2 && \text{(3a)} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} && \text{(3b)} \\ & T\mathbf{x} \in \mathcal{H}_0 && \text{(3c)} \end{aligned}$$

For comparison, alternating optimization uses the projection of previous result $\Pi_{\mathcal{D}}(T\mathbf{x}_0)$ as a *point* proxy,

$$\mathbf{x}_* = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \|T\mathbf{x} - \Pi_{\mathcal{D}}(T\mathbf{x}_0)\|_2 \tag{4a}$$

$$\text{s.t.} \quad A\mathbf{x} = \mathbf{b} \tag{4b}$$

which lacks constraint (3c) of our approach. This additional constraint $T\mathbf{x} \in \mathcal{H}_0$ forces the differentials $T\mathbf{x}$ to lie on a *linear* proxy to $\mathcal{D}$ at $\Pi_D(T\mathbf{x}_0)$. As this linear proxy is usually a good local approximation to the constraint set $\mathcal{D}$, the step of (2) (and equivalently (3)) is often much more efficient than that of (4). Figure 2 shows an illustration of an alternating optimization step versus a step of our algorithm.

The second benefit in introducing (2c) as a single hard linear constraint is that solving the optimization problem (2) can be shown to be as computationally efficient as solving the alternating optimization (4). Indeed, we show that a single sparse prefactorization enables obtaining solutions for (2) using only two back-substitutions, even as $\mathcal{H}_0$ changes. This compares to a single back-substitution required for the alternating optimization (4). This means that an iteration of our algorithm is practically as efficient as alternating optimization, however, its higher order approximation property ensures that a much smaller number of iterations are required for convergence.

Gauss-Newton type methods may require a similar number of iterations to converge in comparison to our proposed approach. On the other hand, they usually require solving a different linear system in each iteration. Linearization performed at each iteration modifies the linear system that needs to be solved in a non-trivial manner. In particular, it does not readily allow for an efficient prefactorization as we propose, thus resulting in an order of magnitude slower iterations. Their overall performance, for the problem discussed in this paper, is therefore inferior. We further discuss, compare and demonstrate the tradeoffs between these methods in Section 5.

**Related work.** The problem of satisfying various shape related constraints has been extensively studied in geometry processing. [Bouaziz et al. 2012] propose a generalized alternating optimization approach, minimizing the distance to a prescribed set of geometric constraints. Similar alternating concepts have been successfully employed for the solution of specific problems; for example, [Sorkine and Alexa 2007; Liu et al. 2008; Chao et al. 2010] for the minimization of the As-Rigid-As-Possible deformation energy, or [Liu et al. 2008] for mesh parameterization. Typically, these methods consider constraints which cannot not be all satisfied simultaneously, hence they aim for solutions which are closest to satisfying them all.

The works of [Tang et al. 2014] and [Balzer and Soatto 2014] have made the observation that the convergence properties of first order methods may be significantly improved by employing Gauss-Newton variants. [Tang et al. 2014] show its advantages for shape manipulation, and [Balzer and Soatto 2014] for normal fields, surface reconstruction and photometric optimization. Gauss-Newton and Levenberg-Marquardt have been extensively studied in the context of general non-linear least squares problems, see [Wright and Nocedal 1999; Lange 2013]. Essentially, they exploit the least squares structure in order to *partially* approximate the Hessian, thereby achieving improved convergence properties.

[Lipman 2012] have studied mapping with bounded distortion differentials in 2D, then generalized to 3D in [Kovalsky et al. 2014]. They optimize convex functionals subject to BD constraints as a sequence of convex conic optimization problems, in turn solved using an interior point solver. [Schüller et al. 2013] have proposed a barrier approach for solving the closely related problem of optimization over non-degenerate orientation preserving maps. Most related to this work is the work of [Aigerman and Lipman 2013] which addresses a similar problem; they, however, generate a sequence of quadratic programs solved using an interior point solver. The main limitation of these approaches is their poor scalability.

## 4 Algorithmic details

Our algorithm for approximating the solution of (1) generates a sequence of approximations $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots$. Given a previous estimate $\mathbf{x}_n$, the procedure for computing $\mathbf{x}_{n+1}$ follows these steps:

1. *Project* the lifted variable $\mathbf{z}_n = T\mathbf{x}_n$ onto $\mathcal{D}$, *i.e.*, compute $\Pi_{\mathcal{D}}(\mathbf{z}_n)$.
2. Form $\mathcal{H}_n$ to be the *hyperplane* orthogonal to the projection vector $\mathbf{z}_n - \Pi_{\mathcal{D}}(\mathbf{z}_n)$.
3. *Optimize* (2) with $(\mathbf{x}_*, \mathbf{x}_0) = (\mathbf{x}_{n+1}, \mathbf{x}_n)$.

Next, we elaborate on each of these steps and provide additional algorithmic details:

**Projection on $\mathcal{D}$.** Projecting the lifted variable $\mathbf{z} = T\mathbf{x}$ onto $\mathcal{D}$, denoted by $\Pi_{\mathcal{D}}(\mathbf{z})$, is straightforward. Since $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_m$ is a cartesian product, the projection is separable and takes the form

$$\Pi_{\mathcal{D}}(\mathbf{z}) = \begin{bmatrix} \Pi_{\mathcal{D}_1}(T_1\mathbf{x}) \\ \vdots \\ \Pi_{\mathcal{D}_m}(T_m\mathbf{x}) \end{bmatrix}, \tag{5}$$

where $\Pi_{\mathcal{D}_j}(T_j\mathbf{x})$ is the (independent) projection of the $j$-th differential $T_j\mathbf{x}$ on its corresponding constraint $\mathcal{D}_j$.

In this paper, the individual projections are all the same $\Pi_{\mathcal{D}_j} = \Pi_{\mathcal{D}^K}$ and have closed-form solutions. [Aigerman and Lipman 2013] provide a characterization of the projection of a single matrix $C \in \mathbb{R}^{d\times d}$ onto the set $\mathcal{D}^K$ of matrices with $K$-bounded distortion: Let $C = U\operatorname{diag}(\sigma_1, \ldots, \sigma_d)V^T$ be the signed-SVD decomposition of $C$ with $\sigma_1 \geq \cdots \geq \sigma_{d-1} \geq |\sigma_d|$. The Euclidean projection of $C$ onto $\mathcal{D}^K$ is given by

$$\Pi_{\mathcal{D}^K}(C) = U\operatorname{diag}(\tau_1, \ldots, \tau_d)V^T,$$

where $\{\tau_i\}$ minimize

$$\min_{\{\tau_i\}} \quad \sum_{i=1}^{d} (\sigma_i - \tau_i)^2 \tag{6a}$$

$$\text{s.t.} \quad \tau_1 \leq K\tau_d \tag{6b}$$

We complement their characterization and provide efficient closed-form algorithms for computing the solutions of (6) in the case of $d = 2$ and $d = 3$ in Appendix A. Consequently, we note that order and non-negativity, $\tau_1 \geq \cdots \geq \tau_d \geq 0$, are implicitly imposed.

We finish with an interesting observation we shall later use for the analysis of the algorithm (proof provided in Appendix B):

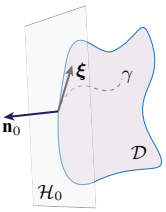
**Lemma 1.** *Projection on $\mathcal{D}$ is contractive, that is, $\|\Pi_{\mathcal{D}}(\mathbf{z})\|_2 \leq \|\mathbf{z}\|_2$. Furthermore, the inequality is strict for nontrivial projections, that is, $\Pi_{\mathcal{D}}(\mathbf{z}) \neq \mathbf{z}$.*

**The proxy hyperplane $\mathcal{H}$.** As a linear proxy we use the hyperplane $\mathcal{H}$ that passes through the projection $\Pi_{\mathcal{D}}(\mathbf{z}_0)$ of the lifted variable $\mathbf{z}_0 = T\mathbf{x}_0$ onto the set $\mathcal{D}$ of BD constraints. We further require it is orthogonal to the direction of projection $\mathbf{n}_0 = \mathbf{z}_0 - \Pi_{\mathcal{D}}(\mathbf{z}_0)$, see Figure 2. Equation (2c) is therefore realized as

$$\mathbf{n}_0^T T\mathbf{x} = \mathbf{n}_0^T \Pi_{\mathcal{D}}(\mathbf{z}_0). \tag{7}$$

As previously mentioned, the hyperplane $\mathcal{H}$ provides a good linear proxy for $\mathcal{D}$ at $\Pi_{\mathcal{D}}(\mathbf{z})$ with tangent-like properties. The following lemma provides a more precise local characterization (see proof in Appendix C).

**Lemma 2.** $\langle \boldsymbol{n}_0, \boldsymbol{\xi} \rangle \leq 0$ *for any one-sided tangent direction* $\boldsymbol{\xi}$ *of* $\mathcal{D}$ *at* $\Pi_{\mathcal{D}}(\mathbf{z}_0)$.




As illustrated in the inset, we define a one-sided tangent direction as the right side derivative at zero $\boldsymbol{\xi} = \gamma'_+(0)$ of a differential curve $\gamma : [0,1] \to \partial\mathcal{D}$ originating at $\Pi_{\mathcal{D}}(\mathbf{z}_0)$, that is, $\gamma(0) = \Pi_{\mathcal{D}}(\mathbf{z}_0)$.

**Efficient linear step.** Each iteration of (2) requires solving a linearly constrained least squares problem. In turn, this can be shown to be equivalent to solving the following KKT linear system (*e.g.*, see [Boyd and Vandenberghe 2004]):

$$\begin{bmatrix} T^TT & A^T & T^T\mathbf{n}_0 \\ A & 0 & 0 \\ \mathbf{n}_0{}^TT & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \mu \end{bmatrix} = \begin{bmatrix} T^TT\mathbf{x}_0 \\ \mathbf{b} \\ \mathbf{n}_0{}^T\Pi_{\mathcal{D}}(\mathbf{z}_0) \end{bmatrix}. \tag{8}$$

Had the left-hand-side of (8) been fixed, the solution at each iteration could have been obtained using prefactorization. This is, for example, the case in standard alternating approaches. In our case, however, the left-hand-side of (8) is updated in each iteration, as the projection normal $\mathbf{n}_0$ is recomputed; thus straightforward prefactorization cannot be used.

We exploit the specific structure of (8), which can be rewritten as

$$\begin{bmatrix} M & \boldsymbol{\eta}_0 \\ \boldsymbol{\eta}_0{}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{c}_0 \\ d_0 \end{bmatrix}, \tag{9}$$

where

$$M = \begin{bmatrix} T^TT & A^T \\ A & 0 \end{bmatrix}, \quad \boldsymbol{\eta}_0 = \begin{bmatrix} T^T\mathbf{n}_0 \\ 0 \end{bmatrix},$$

and $[\mathbf{c}_0; d_0]$ is the respective splitting of the right-hand-side of (8).

Since $M$ is fixed during the iterations it can be prefactorized (*e.g.*, using sparse $LU$ or $LDL^T$); moreover, for meshes, $M$ has a Laplacian-like sparsity pattern, thus the factorization is efficiently computed resulting in highly sparse factors.

Using a factorization of $M$, a linear system of the form $M\mathbf{y} = \mathbf{c}$ can be solved efficiently using back-substitution. We let $F_M(\mathbf{c})$ denote the solution corresponding to such an equation. The derivation in Appendix D shows that the solution to equation (8) is given by

$$\begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \mathbf{y}_{\mathbf{c}} - \frac{\boldsymbol{\eta}_0{}^T\mathbf{y}_{\mathbf{c}} - d_0}{\boldsymbol{\eta}_0{}^T\mathbf{y}_{\boldsymbol{\eta}}}\mathbf{y}_{\boldsymbol{\eta}}, \tag{10}$$

where $\mathbf{y}_{\mathbf{c}} = F_M(\mathbf{c}_0)$ and $\mathbf{y}_{\boldsymbol{\eta}} = F_M(\boldsymbol{\eta}_0)$.

Note that this implies that the computational complexity of solving the linear part of the algorithm, *i.e.*, Equation (2), is that of performing *only two* back-substitutions using the factorization of $M$, namely, $F_M(\mathbf{c}_0)$ and $F_M(\boldsymbol{\eta}_0)$.

**Non-singularity of the KKT system.** Next we show that the algorithm can always find a unique solution to (2).

**Theorem 1.** *If $A$ is full-rank and determines global translation, and if $\boldsymbol{n}_0 \neq 0$ (i.e., current estimate $\mathbf{x}_0$ is infeasible) then the KKT matrix in* (8) *is invertible.*

By determining global translation we mean that if $A\mathbf{x} = b$ then global translations of $\mathbf{x}$ (with respect to the coordinates it represents in $\mathbb{R}^d$) fail to obey the same constraint. More formally, global translations take the form $\mathbf{x} + \mathbf{t}$, where $\mathbf{t}$ is defined by

$$\mathbf{t} = \mathrm{vec}\left(\mathrm{diag}\,(t_1, \ldots, t_d)\,\mathbf{1}_d\mathbf{1}_n{}^T\right), \tag{11}$$

with $\mathbf{1}_d \in \mathbb{R}^d, \mathbf{1}_n \in \mathbb{R}^n$ the all ones vectors. The condition on $A$ simply means that all nontrivial $\mathbf{t}$'s *do not* belong to the null space of $A$. Hence, $A(\mathbf{x} + \mathbf{t}) \neq b$ for $\mathbf{t} \neq 0$. This condition holds in many standard cases; for example, if $A$ includes at least one point positional constraint or fixes the centroid of some vertex positions.

*Proof.* First, we show that $M$ is non-singular. Since $A$ is full rank, a necessary and sufficient condition for the invertibility of $M$ is that $T^TT$ and $A$ have no nontrivial common null space (see [Boyd and Vandenberghe 2004], page 523). The matrix $T^TT$ is the Laplacian of the mesh (up to scaling by element masses). Therefore, its null space is spanned by the coordinate-constants, which are exactly the vectors $\mathbf{t}$ as in (11). This means that $T^TT$ and $A$ have no nontrivial common null space.

Since $M$ is invertible, it suffices to show that $\mathbf{n}_0{}^TT \neq 0$. By the definitions of $\mathbf{n}_0$ and $\mathbf{z}_0$ we have that

$$\mathbf{n}_0{}^TT\mathbf{x}_0 = (\mathbf{z}_0 - \Pi_{\mathcal{D}}(\mathbf{z}_0))^T\mathbf{z}_0 = \|\mathbf{z}_0\|_2^2 - \mathbf{z}_0{}^T\Pi_{\mathcal{D}}(\mathbf{z}_0).$$

Applying the Cauchy-Schwarz inequality to the last term gives

$$\left|\mathbf{z}_0{}^T\Pi_{\mathcal{D}}(\mathbf{z}_0)\right| \leq \|\mathbf{z}_0\|_2 \left\|\Pi_{\mathcal{D}}(\mathbf{z}_0)\right\|_2.$$

Using the contraction property of $\Pi_{\mathcal{D}}$, Lemma 1, shows that

$$\left|\mathbf{z}_0{}^T\Pi_{\mathcal{D}}(\mathbf{z}_0)\right| < \|\mathbf{z}_0\|_2^2.$$

Thus, we have that $\mathbf{n}_0{}^TT\mathbf{x}_0 > 0$ which concludes the proof. □

**Generalized metric.** One may prefer a different metric for the objective of (2). For instance, in the case of simplicial complexes a natural choice is to replace the functional with

$$\|T\mathbf{x} - T\mathbf{x}_0\|_W^2 = (T\mathbf{x} - T\mathbf{x}_0)^T W (T\mathbf{x} - T\mathbf{x}_0).$$

where $W$ is a diagonal matrix accounting for element masses (areas or volumes). This choice better respects the intrinsic geometry and lessens the effects of discretization. Such weighting can be straightforwardly incorporated into the algorithm and derivations previously presented.

**Termination, convergence and feasibility.** Theorem 1 asserts that the algorithm iterations are well defined provided that the projection direction, $\mathbf{n}_0$, does not vanish. Moreover, note that $\mathbf{n}_0 = T\mathbf{x}_0 - \Pi_{\mathcal{D}}(T\mathbf{x}_0) = 0$ implies feasibility for Problem (1). Therefore, we use a threshold on the magnitude of $\mathbf{n}$ to determine termination. The solution upon termination is thus guaranteed to be feasible, although the algorithm is not guaranteed to terminate. Nevertheless, the proposed method performs well in practice, as demonstrated in the next sections.

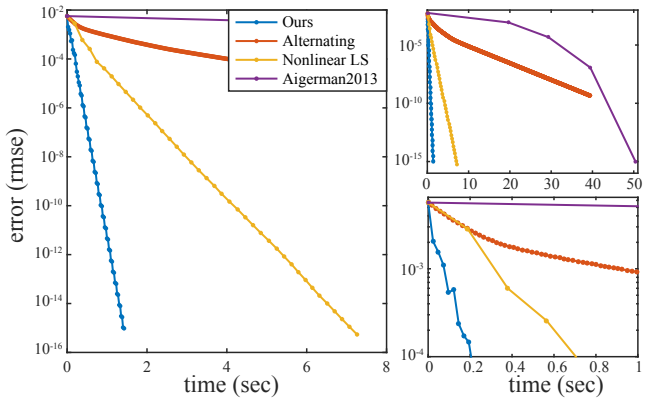



**Figure 3:** *Comparison of convergence rates. Typical convergence behavior obtained for a mapping of a tetrahedral mesh with 13k elements. Vertical axes show the magnitude of $\boldsymbol{n}_0 = \mathbf{z}_0 - \Pi_{\mathcal{D}}(\mathbf{z}_0)$, quantifying the violation of the constraint $\mathbf{z}_0 \in \mathcal{D}$. Dots indicate the iterations of the different methods. Additional graphs present a longer time-range (top) and zoom-in on the first second (bottom).*

An implicit assumption is that the distortion bound $K$ is prescribed such that Problem (1) has a feasible solution. Otherwise, the termination criterion above is never met and the algorithm fails to terminate. Choosing a feasible $K$, or determining its existence, is an open problem.

## 5 Evaluation

In this section we evaluate the performance and scalability of the proposed approach. We compare our algorithm with three closely related alternative methods for approximating the solution of problem (1):

**Alternating optimization** adopting the approach of [Bouaziz et al. 2012]. As described in (4), the projected estimate of previous iteration, $\Pi_{\mathcal{D}}(T\mathbf{x}_0)$, is used as a point proxy for the constraint $T\mathbf{x} \in \mathcal{D}$. Each iteration comprises projection onto $\mathcal{D}$ and the solution of a linear system with fixed left-hand-side. Arguably, this approach sets a lower bound on the computational cost per-iteration.

**Non-linear least-squares** adopting the approach of [Tang et al. 2014]. First, we replace the constraint $T\mathbf{x} \in \mathcal{D}$ with a soft-constraint, measuring the amount by which each differential, $T_j\mathbf{x}$, fails to satisfy the bounded distortion constraint $T_j\mathbf{x} \in \mathcal{D}^K$. This yields the minimization,

$$\underset{A\mathbf{x}=b}{\operatorname{argmin}} \ \|T\mathbf{x} - T\mathbf{x}_0\|_2^2 + \sum_{j=1}^{m} \lambda_j \left\|T_j\mathbf{x} - \Pi_{\mathcal{D}^K}(T_j\mathbf{x})\right\|_2^2 .$$

Then, each term of the latter sum is linearized with respect to the estimate of previous iteration, $\mathbf{x}_0$, resulting with

$$\underset{A\mathbf{x}=b}{\operatorname{argmin}} \ \|T\mathbf{x} - T\mathbf{x}_0\|_2^2 + \sum_{j=1}^{m} \lambda_j \left\langle \mathbf{n}_0^j, T_j\mathbf{x} - \Pi_{\mathcal{D}^K}(T_j\mathbf{x}_0)\right\rangle^2 ,$$

where $\mathbf{n}_0^j = T_j\mathbf{x}_0 - \Pi_{\mathcal{D}^K}(T_j\mathbf{x}_0)$ is the projection direction of the $j$-th differential.

Note that attempting to realize the individual linearizations as hard constraints, in an analogous manner to our hyperplane proxy (2c), is bound to fail. Empirically, the linear system is infeasible with $\left\langle \mathbf{n}_0^j, T_j\mathbf{x} - \Pi_{\mathcal{D}^K}(T_j\mathbf{x}_0)\right\rangle = 0$ imposed.

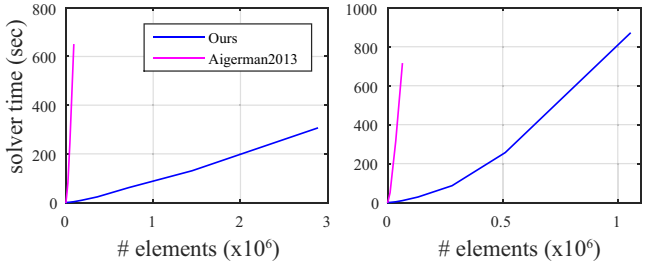



**Figure 4:** *Scalability of the proposed algorithm. Typical overall running time as a function of problem size for 2D (left) and 3D (right) mappings.*

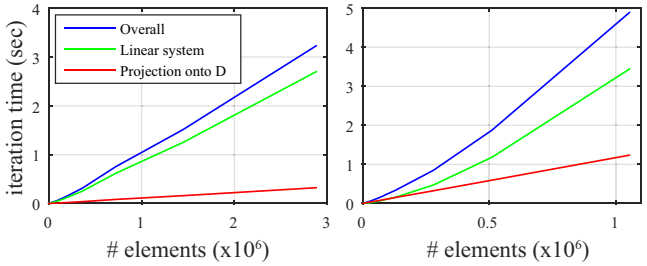



**Figure 5:** *Iteration time as a function of problem size for 2D (left) and 3D (right) problems. The graphs further show the time spent on solving the linear (green) and projection (red) parts of the algorithm.*

**Aigerman2013** also approximate the projection of mappings on bounded distortion differentials [Aigerman and Lipman 2013]. They propose an iterative procedure whereby each differential constraint, $T_j\mathbf{x} \in \mathcal{D}^K$, is replaced with a linear half-space. This leads to a sequence of quadratic programs, in turn solved using an interior point solver.

Figure 3 shows typical convergence behavior of these approaches, obtained for a 13k tetrahedral mesh. Error is plotted against time, whereas dots indicate the iterations of each of the methods. Both the alternating and the proposed approach enjoy speedup achieved via a one-time prefactorization, taking 100 millisecond for this example. The iterations of the alternating approach are fastest, at 20 millisecond per iteration; its linear part requires a single back-substitution using the factorization of $M$; its convergence rate, however, is poor. The iterations of the proposed approach are almost as fast, at 24 millisecond per iteration, requiring only two back-substitutions as in (10); it converges first in under 60 iterations. The non-linear least squares approach converges in 40 iterations; each iteration, however, requires solving a different linearly constrained least-squares problem, resulting in 188 milliseconds per iteration. Typically, the iterations of non-linear least squares are 10-100 times longer than the iterations of our approach (varying with problem size and the number of violated constraints at a specific iteration). Moreover, it requires non-trivial tuning of the weights $\lambda_1, \ldots, \lambda_m$, which were manually fine-tuned for this specific example; this choice of parameters is rather sensitive and incorrect setting leads to either poor convergence rate or complete failure. Aigerman2013 converges most efficiently in only 4 iterations; however, the use of an interior-point solver leads to significantly slower overall performance, averaging at about 12.5 seconds per iteration.

Figure 4 demonstrates the scalability of the approach. It show overall running time (including setup, prefactorization and iterative procedure) as a function of problem size. Timings are presented for 2- and 3-dimensional mappings with over a million elements. Respective running times of Aigerman2013 are presented for comparison.

| | | | Ours | | | [Aigerman and Lipman 2013] | | | [Kovalsky et al. 2014] | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | # vert | # elem | # iter | time (sec) | energy (optimality) | # iter | time (sec) | energy (optimality) | # iter | time (sec) | energy |
| 1LSCM | 85 | 128 | 72 | **0.04** | 0.165 (99.2%) | 4 | 0.80 | 0.168 (97.5%) | 2 | 0.44 | 0.164 |
| 2LSCM | 85 | 128 | 544 | **0.29** | 0.587 (98.8%) | 16 | 2.61 | 0.589 (98.4%) | 2 | 0.50 | 0.580 |
| 3LSCM | 81 | 128 | 66 | **0.03** | 0.114 (99.5%) | 5 | 0.94 | 0.118 (96.8%) | 2 | 0.45 | 0.114 |
| 4LSCM | 81 | 128 | 77 | **0.04** | 0.434 (98.6%) | 7 | 1.25 | 0.438 (97.7%) | 2 | 0.59 | 0.428 |
| 2d_dino | 301 | 484 | 21 | **0.02** | 41.147 (95.0%) | 5 | 1.92 | 40.566 (96.3%) | 2 | 2.05 | 39.070 |
| bar_bend_5 | 4840 | 23400 | 16 | **1.01** | 12.354 (93.8%) | 5 | 127.99 | 14.152 (81.9%) | 3 | 296.52 | 11.589 |
| bar_move_5 | 4840 | 23400 | 13 | **0.85** | 4.393 (92.0%) | 4 | 104.03 | 4.327 (93.4%) | 3 | 297.76 | 4.043 |
| bar_twist_5 | 4840 | 23400 | 9 | **0.71** | 1.990 (93.9%) | 3 | 95.87 | 1.925 (97.1%) | 2 | 209.86 | 1.868 |
| elephant_down_5 | 1933 | 7537 | 8 | **0.18** | 0.067 (87.3%) | 4 | 31.19 | 0.074 (79.9%) | 2 | 67.41 | 0.059 |
| elephant_forward_5 | 1933 | 7537 | 24 | **0.42** | 0.044 (88.6%) | 4 | 32.31 | 0.047 (82.3%) | 2 | 73.63 | 0.039 |
| elephant_stand_5 | 1933 | 7537 | 7 | **0.17** | 0.078 (88.9%) | 4 | 30.87 | 0.086 (80.6%) | 2 | 76.56 | 0.070 |
| gorilla_from_arap_5 | 5269 | 20765 | 10 | **0.59** | 0.010 (83.0%) | 3 | 78.40 | 0.009 (93.8%) | 2 | 263.96 | 0.008 |
| biharmonic_7 | 2002 | 9242 | 125 | **2.11** | 30.669 (94.5%) | 6 | 49.80 | 31.640 (91.6%) | 3 | 137.44 | 28.972 |
| arma_mvc_large8 | 14868 | 47761 | 9 | **1.62** | 57.779 (83.9%) | 5 | 290.76 | 50.509 (96.0%) | 2 | 497.28 | 48.486 |
| arma_mvc_large9 | 14868 | 47761 | 9 | **1.66** | 42.621 (85.6%) | 5 | 301.02 | 37.448 (97.4%) | 2 | 479.05 | 36.483 |
| bimba_polycube8 | 10790 | 45422 | 44 | **4.73** | 0.170 (84.7%) | 6 | 398.40 | 0.158 (91.1%) | 2 | 556.82 | 0.144 |
| duck_cube4 | 2464 | 12601 | 81 | **2.21** | 0.344 (94.4%) | 7 | 105.41 | 0.338 (95.9%) | 2 | 139.10 | 0.325 |
| hand_polycube20 | 8366 | 40627 | 130 | **11.29** | 0.051 (69.9%) | 11 | 630.83 | 0.074 (48.4%) | 2 | 512.10 | 0.036 |
| maxplanck_cube6 | 9867 | 40076 | 41 | **3.81** | 0.155 (87.8%) | 11 | 590.60 | 0.137 (99.3%) | 2 | 525.51 | 0.136 |
| rocker_polycube20 | 12428 | 60301 | 172 | **22.09** | 0.079 (85.7%) | 10 | 1008.03 | 0.096 (70.5%) | 2 | 784.41 | 0.068 |
| sphinx_polycube10 | 10528 | 43371 | 42 | **4.27** | 0.061 (84.9%) | 14 | 823.20 | 0.063 (82.4%) | 2 | 643.45 | 0.052 |
| | | | | | Avg. **90.0%** | | | Avg. **89.0%** | | | |

**Table 1:** *Comparing our approach with [Aigerman and Lipman 2013] and [Kovalsky et al. 2014]. Our approach achieves comparable results to [Aigerman and Lipman 2013] at a fraction of the time – 100 times faster on average. [Kovalsky et al. 2014] directly optimize Problem* (1)*, but require a feasible initialization, provided here by our approach; although initialized with a near optimal point their runtime is high. To measure optimality we present the energy ratio with respect to [Kovalsky et al. 2014]; averages shown at the bottom.*

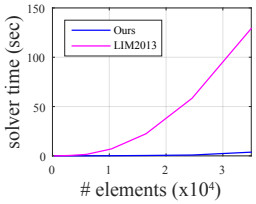
Figure 5 further profiles the iterations of the proposed approach, showing the amount of time spent on solving the linear (green) and projection (red) parts of the algorithm. Since projection onto $\mathcal{D}$ separates into independent differential projections, equation (5), it scales linearly with problem size. The algorithm is implemented in MATLAB. The projection onto $\mathcal{D}$ is implemented in a sequential single-thread C function; further speedup may be achieved via parallelization. All timings were measured on a 3.50GHz Intel i7.

**Comparison with [Aigerman and Lipman 2013] and [Kovalsky et al. 2014].** We have compared the performance of the proposed approach with that of [Aigerman and Lipman 2013] on the entire dataset of small to medium scale problems provided by the authors. We further used the method of [Kovalsky et al. 2014] to directly compute the projection onto the set of bounded distortion mappings, Problem (1); we used the output of the proposed approach to provide the latter with the feasible initialization it requires.

Table 1 summarizes this comparison. On average, the proposed algorithm runs 100 times faster than Aigerman's method; and more than 150 times faster than Kovalsky's method (MOSEK optimization time presented, excluding setup time), even though it is near-optimally initialized with the feasible results obtained using our approach. Kovalsky achieves the lowest energies, as it directly minimizes the projection energy, $\|T\mathbf{x} - T\mathbf{x}_0\|_2$. Our approach and [Aigerman and Lipman 2013] achieve comparable results, attaining about $90\%$ average energy ratio with respect to Kovalsky's results.

**Comparison with LIM [Schüller et al. 2013].** This work devises a barrier method for optimizing mappings that avoid flipped elements, by directly imposing $\det C_j > \varepsilon$ on the differentials.

We compare LIM to the following approach: first, optimize for the mapping, without enforcing non-flip constraints; then, in a second step, find a similar $K$-bounded distortion map with a high constant (*e.g.*, $K = 1e4$). This is motivated by the observation that such maps provide a natural scale-invariant characterization of mappings free of flipped elements. (We note, however, that local injectivity is not immediately implied.) In turn, this enables approximating large-scale problems where LIM requires excessive time or fails to run. The inset compares the scalability of LIM to that of the proposed approach for moderately sized problems. (LIM timings were obtained using code provided by the authors.)



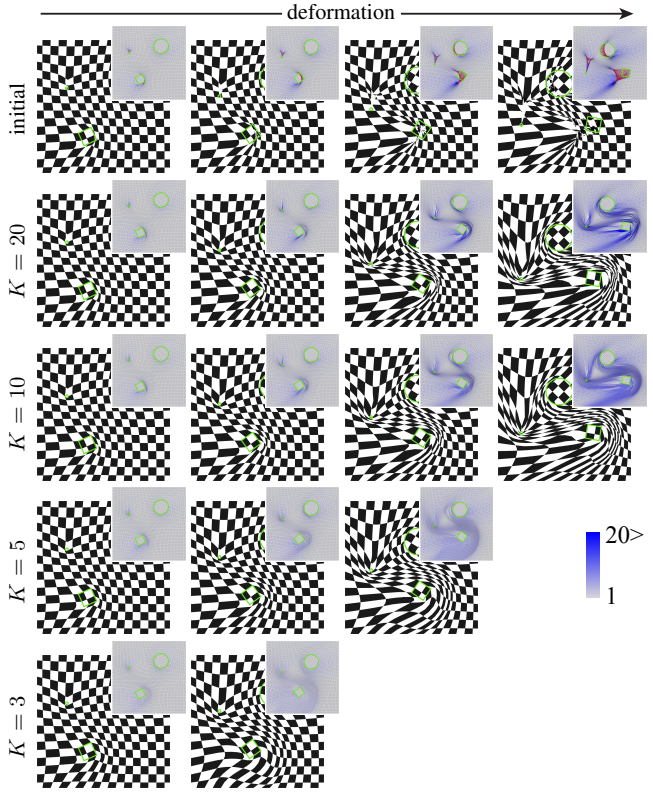




**Figure 6:** *Robustness of the proposed algorithm. Top row shows initial maps obtained by minimizing the Dirichlet energy subject to increasingly translating and rotating a square, disk and a point (illustrated in green). The insets depict conformal distortions (blue shades) and flipped triangles (red edges). Bottom rows show the result of applying the proposed approach with decreasing distortion bounds, $K$. Absent results (bottom-right) indicate cases for which the algorithm failed to converge (possibly due to infeasibility).*

**Robustness and failure cases.** Figure 6 further demonstrates the robustness of the proposed approach and presents failure cases. It shows the result of employing the algorithm with varying distortion bounds to initial mappings undergoing increasing level of deformation. The algorithm successfully handles challenging cases, but fails to converge for the most extreme initial mappings at low distortion bounds.

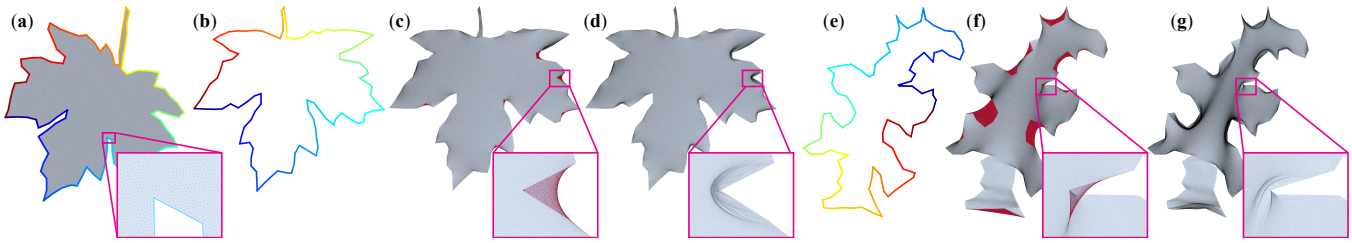



**Figure 7:** *Mapping of 2D shapes triangulated with 326k triangles. The source (a) is mapped into two different target domains (b) and (f). For the first target, (c) and (d) show the initial map and bounded distortion map ($K = 20$) with fixed boundary. For the second extremely challenging target, (f) and (g) show the initial map and bounded distortion map ($K = 100$).*

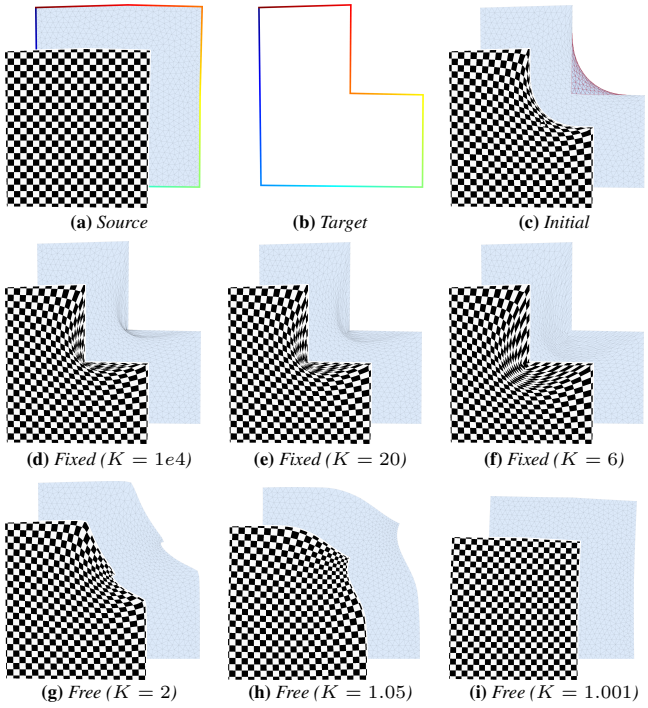



**Figure 8:** *Illustrative example of 2D mappings of domains. First row shows the source (a) and target (b) domains; the colored outline illustrates the prescribed mapping of the boundaries. (c) shows the initial map obtained by minimizing the Dirichlet energy; red edges indicate flipped triangles. Second and third rows show bounded distortion maps obtained with the proposed approach for different bounds $K$. In the second row the boundary is fixed, while in the third row the boundary is free so as to enable imposing lower bounds on distortion.*

## 6 Experiments

### 6.1 Large-scale mappings

**Mapping 2D domains.** In this experiment we compute mappings between 2D domains. Figure 8 illustrates the setup of this experiment. We are given a pair of parameterized closed planar curves describing the boundaries of 2D domains (Figures 8a and 8b); the colored outline illustrates a prescribed mapping of the boundaries. Mapping a triangulation of the source onto the target, *e.g.*, by minimizing the Dirichlet energy, usually results in flipped or nearly-degenerate triangles (see 8c).

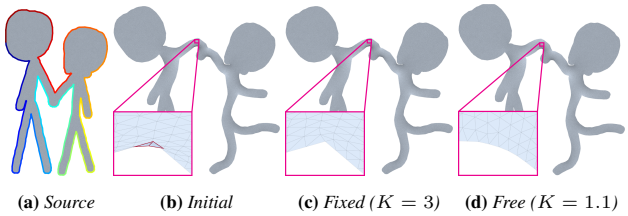



**Figure 9:** *Mapping of 2D shapes comprising 218k triangles. The initial map has flipped and nearly-degenerate triangles (b). The proposed algorithm achieves a map with bounded distortion $K = 3$ in 2.8 seconds (c). Releasing the boundary (d) enables lowering the distortion to $K = 1.1$, without resulting in a significant modification to the target shape.*

The proposed algorithm enables approximating this initial map with similar better-behaved maps. Setting different bounds on distortion $K$ enables controlling the amount of allowed local change in aspect ratio induced by the map. The approximating map can be forced to respect the boundary constraints, via $A\mathbf{x} = b$, see Figures 8d-8f. If the distortion bound is set too low for a specific problem, the boundary constraints cannot be fully satisfied (as the problem becomes infeasible); in which case, a mapping with free boundary constraints may be computed, see Figures 8g-8i.

Figure 9 demonstrates the mapping of a domain comprising 218k triangles. The algorithm achieves a bounded distortion map with constant $K = 3$ in 2.8 seconds. Releasing the boundary allows further lowering the distortion, achieving a bounded distortion map with $K = 1.1$. The proposed algorithm naturally regularizes for deviations in differentials, minimizing $\|T\mathbf{x} - T\mathbf{x}_0\|_2$ with respect to previous iteration. Thus, only mild changes to the target shape occur, accommodating the lower bound on distortion.

A more challenging problem is demonstrated in Figure 7, where a domain comprising 326k triangles is mapped into two different target shapes (curves taken from [Telea and Van Wijk 2002]). The initial mappings suffer from poor behavior near the boundaries, with 1.6% and 12.3% flipped or near-degenerate triangles, respectively shown in Figures 7c,f. In both cases the proposed algorithm achieves a bounded distortion map, thereby, inducing a global bijection between the shapes.

**Mapping of 3D volumes.** Figures 1 and 10 show examples of volumetric mappings obtained with the proposed algorithm. The models comprise 1.2M and 678k tetrahedra, respectively. For each model, the initial mapping was obtained by minimizing the Dirichlet energy subject to a prescribed boundary map, resulting in about 10% near-degenerate or flipped tets. In both challenging cases we have achieved a fixed-boundary bounded distortion map with a constant of $K = 50$, inducing a global bijection between the models.

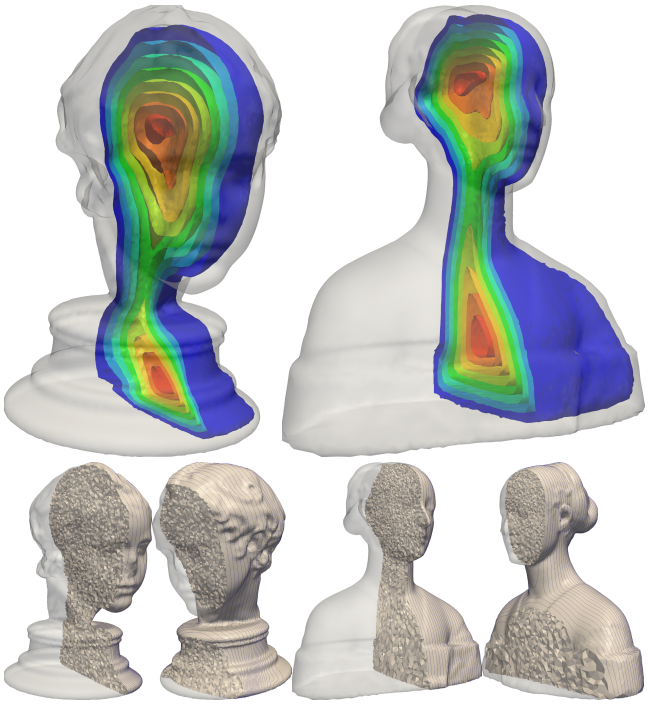


**Figure 10:** *Bijective volumetric mapping of models comprising 678k tetrahedra. The proposed algorithm is used to compute a fixed-boundary bounded distortion mapping. The mapping is depicted via corresponding isocontours (top) and two corresponding sections through the volume (bottom).*

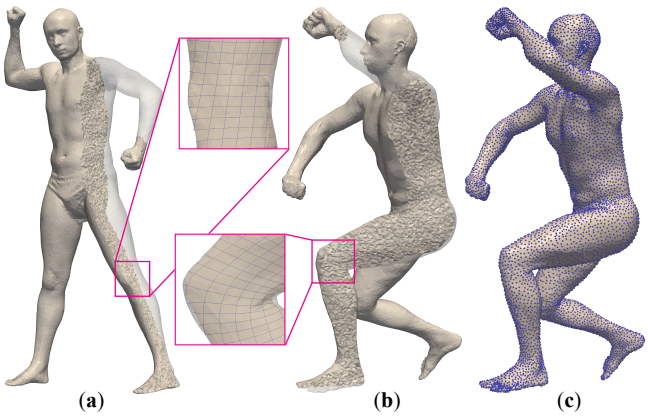



**Figure 11:** *Low distortion mapping of 3D volumes obtained using the proposed algorithm. Releasing the boundary enables computing a $K = 5$ bounded distortion mapping (b), with minor changes to the boundary. The target boundary vertices (purple) are overlaid on the resulting boundary surface in (c).*

Figure 11 shows the volumetric mapping of two SCAPE models [Anguelov et al. 2005], comprising 810k tetrahedra. In this case, a low distortion map might be expected, as the objects undergo a nearly-isometric deformation. However, poor initial boundary mapping renders the task of volumetric mapping highly challenging. Nevertheless, as Figure 11c demonstrates, releasing the boundary allows obtaining a high quality map ($K = 5$), without a significant sacrifice of accuracy on the boundary.

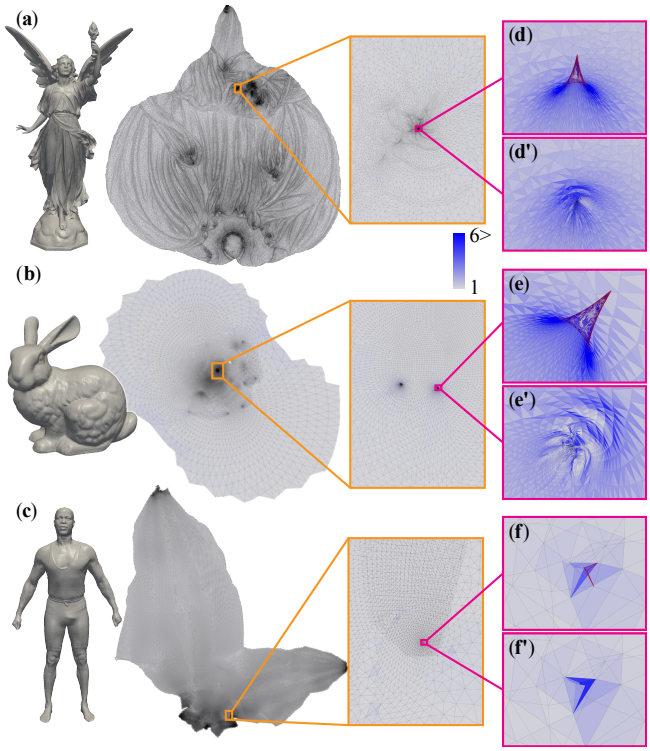



**Figure 12:** *Bounded conformal distortion for large scale parameterization (525k, 69k and 293k triangles). Minimizing the LSCM energy results in parameterizations that suffer from high conformal distortion and flipped triangles, (d)-(f); blue shades quantify conformal distortions and red edges indicate flipped triangles. Bounded conformal distortion parameterization ($K = 5$) are obtained using the proposed algorithm, (d')-(f').*

### 6.2 Large-scale parameterization with bounded conformal distortion

Parameterization (flattening) of large-scale surfaces is a standing problem in geometry processing. Some methods aiming at conformal parameterization rely on the solution of a sparse linear system [Lévy et al. 2002] or eigen-decomposition [Mullen et al. 2008], and scale reasonably to large problems. However, they often result with mappings having some flipped triangles or triangles with high conformal distortion.

Using the proposed algorithm for finding a similar bounded distortion parameterization is straightforward. Figure 12 shows LSCM parameterizations of three models, comprising 525k, 69k and 293k triangles, respectively, of which, $1.1\%$, $5.6\%$ and $0.1\%$ are either flipped or suffer a high distortion; examples are shown in the blowups (d)-(f). Employing our algorithm results in bounded distortion maps with constant $K = 5$, as shown in (d')-(f').

### 6.3 Interactive rate bounded distortion maps

So far, we have discussed large-scale problems, where other approaches for computing maps of bounded distortion either scale poorly or fail altogether. Small to medium scale problems may also benefit from the proposed algorithm, which enables computation of bounded distortion mappings at interactive rates.

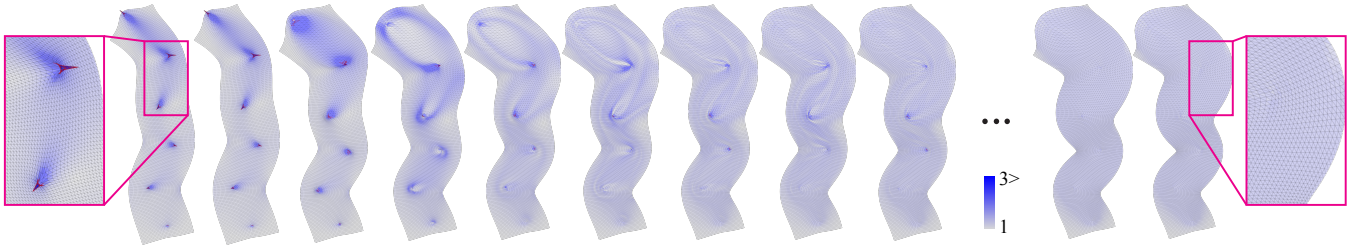



**Figure 13:** *Interactive rate bounded distortion deformation – showing the iterations of the proposed algorithm. An initial map is obtained by minimizing the ARAP energy subject to point constraints (left); blue shades quantify conformal distortions and red edges indicate flipped triangles. The algorithm converges in 30 iterations to a bounded distortion map with $K = 1.5$ (right). Each iteration takes 10 millisecond for this 12k triangles mesh.*

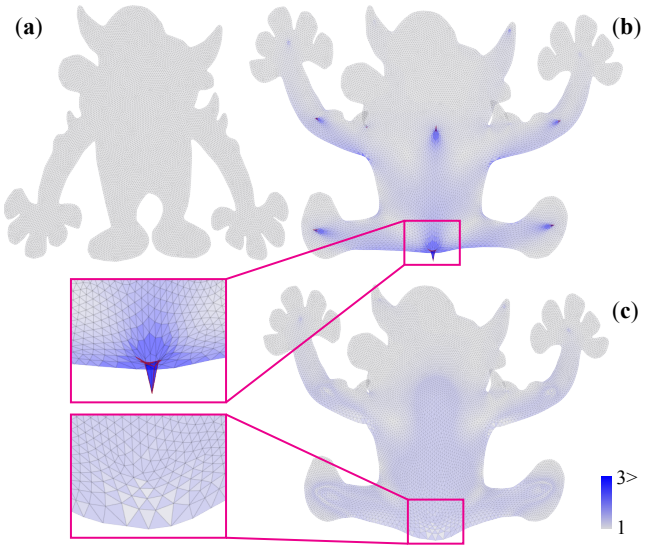



**Figure 14:** *Interactive rate bounded distortion deformation. A monster model (a) is deformed by minimizing the ARAP energy subject to point constraints (b); blue shades quantify conformal distortions and red edges indicate flipped triangles. Our algorithm produces a similar bounded distortion map with $K = 1.5$ in 20 iterations, converging within 0.25 seconds.*

Figure 13 illustrates the iterations of the proposed algorithm (left to right). The left shows a bar comprising 12k triangles, deformed by minimizing the ARAP energy subject to moving point handles [Igarashi et al. 2005; Sorkine and Alexa 2007; Liu et al. 2008; Chao et al. 2010]. Given this initial map, the algorithm produces the bounded distortion map shown in the right ($K = 1.5$) in only 30 iterations, at an average time of 10 millisecond per iteration. Visually plausible results are obtained after just a few iterations, thus enabling user interaction. Figure 14 shows a more elaborate ARAP deformation of a monster shape comprising 23k triangles. In this case, our algorithm has converged in 20 iterations, each taking an average of 13 millisecond.

## 7 Concluding remarks

We have proposed an algorithm for computing large-scale bounded distortion maps of triangular and tetrahedral meshes. Our approach has comparable computational efficiency to that of alternating optimization, yet has improved convergence properties typically associated with higher order methods. Key to our approach is the use of a single linear hyperplane, updated at each iteration, providing a local approximation to the set of bounded distortion maps.

As typical to highly non-linear and non-convex problems such as ours, the algorithm we propose lacks global convergence guarantees. Nevertheless, it performs well on various problems, as we demonstrate in the paper. Its current form and complementing theory are currently limited to mappings strictly satisfying bounds on distortion. Future directions include extending the algorithm to minimize other energy functionals, other types of constraints, and to address the common case of infeasible constraints (*e.g.*, a low distortion bound $K$).

## 8 Acknowledgements

This work was supported in part by the European Research Council (ERC starting grant No. 307754 "SurfComp"), the Israel Science Foundation (grant No. 1284/12 and 1265/14) and the I-CORE program of the Israel PBC and ISF (Grant No. 4/11). The meshes used are from SHREC07 dataset [Giorgi et al. 2007], SCAPE dataset [Anguelov et al. 2005], and Stanford 3D Scanning Repository. The authors would like to thank the anonymous reviewers for their helpful comments and suggestions.

## References

AIGERMAN, N., AND LIPMAN, Y. 2013. Injective and bounded distortion mappings in 3d. *ACM Trans. Graph. 32*, 4, 106–120.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. Scape: Shape completion and animation of people. *ACM Trans. Graph. 24*, 3, 408–416.

BALZER, J., AND SOATTO, S. 2014. Second-order shape optimization for geometric inverse problems in vision. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, IEEE, 3850–3857.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 1657–1667.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.

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

GIORGI, D., BIASOTTI, S., AND PARABOSCHI, L., 2007. Shape retrieval contest 2007: Watertight models track.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (July), 1134–1141.

KOVALSKY, S. Z., AIGERMAN, N., BASRI, R., AND LIPMAN, Y. 2014. Controlling singular values with semidefinite programming. *ACM Trans. Graph. 33*, 4 (July), 68:1–68:13.

LANGE, K. 2013. *Optimization (Springer Texts in Statistics)*, 2nd ed. 2013 ed. Springer, 3.

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

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

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. J. 2008. A local/global approach to mesh parameterization. *Proc. Eurographics Symposium on Geometry Processing 27*, 5.

MULLEN, P., TONG, Y., ALLIEZ, P., AND DESBRUN, M. 2008. Spectral conformal parameterization. In *Computer Graphics Forum*, vol. 27, Wiley Online Library, 1487–1494.

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. *Proc. Eurographics Symposium on Geometry Processing 32*, 5, 125–135.

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

TANG, C., SUN, X., GOMES, A., WALLNER, J., AND POTTMANN, H. 2014. Form-finding with polyhedral meshes made simple. *ACM Trans. Graph. 33*, 4, 70.

TELEA, A., AND VAN WIJK, J. J. 2002. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the symposium on Data Visualisation 2002*, Eurographics Association.

WRIGHT, S. J., AND NOCEDAL, J. 1999. *Numerical optimization*, vol. 2. Springer New York.

## Appendix A Bounded distortion projection

**2-dimensional case.** Given $\sigma_1 \geq |\sigma_2|$ we compute

$$\min_{\tau_1,\tau_2} \quad (\sigma_1 - \tau_1)^2 + (\sigma_2 - \tau_2)^2 \tag{12a}$$

$$\text{s.t.} \quad \tau_1 \leq K\tau_2 \tag{12b}$$

If $\sigma_1 \leq K\sigma_2$ then the input is already bounded distortion with constant $K$. Otherwise, if $\sigma_1 > K\sigma_2$, the solution of (12) must satisfy $\tau_1 = K\tau_2$. Hence $(\tau_1, \tau_2) = (Kt, t)$, where $t$ is the minimizer of

$$\min_t \quad (\sigma_1 - Kt)^2 + (\sigma_2 - t)^2, \tag{13}$$

attained by $t = (K\sigma_1 + \sigma_2)/(1 + K^2)$.

**3-dimensional case.** Given $\sigma_1 \geq \sigma_2 \geq |\sigma_3|$ we compute

$$\min_{\tau_1,\tau_2,\tau_3} \quad (\sigma_1 - \tau_1)^2 + (\sigma_2 - \tau_2)^2 + (\sigma_3 - \tau_3)^2 \tag{14a}$$

$$\text{s.t.} \quad \tau_1 \leq K\tau_3 \tag{14b}$$

If $\sigma_1 \leq K\sigma_3$ then the input is already bounded distortion with constant $K$. Otherwise, $\sigma_1 > K\sigma_3$ thus the solution of (14) must satisfy $\tau_1 = K\tau_3$. First, assume $\tau_2 = \sigma_2$ and let $(\tau_1, \tau_2, \tau_3) = (Kt, \sigma_2, t)$ where $t = (K\sigma_1 + \sigma_3)/(1 + K^2)$. If $Kt \geq \sigma_2 \geq t$ we are done.

Otherwise, if $Kt < \sigma_2$ the solution of (14) must satisfy $\tau_1 = \tau_2 = K\tau_3$. Hence, $(\tau_1, \tau_2, \tau_3) = (Kt, Kt, t)$, where $t$ is the minimizer of

$$\min_t \quad (\sigma_1 - Kt)^2 + (\sigma_2 - Kt)^2 + (\sigma_3 - t)^2, \tag{15}$$

which is attained by $t = (K\sigma_1 + K\sigma_2 + \sigma_3)/(1 + 2K^2)$.

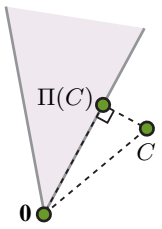
If $\sigma_2 < t$ the solution of (14) must satisfy $\tau_1 = K\tau_2 = K\tau_3$. Hence, $(\tau_1, \tau_2, \tau_3) = (Kt, t, t)$, where $t$ is the minimizer of

$$\min_t \quad (\sigma_1 - Kt)^2 + (\sigma_2 - t)^2 + (\sigma_3 - t)^2, \tag{16}$$

which is attained by $t = (K\sigma_1 + \sigma_2 + \sigma_3)/(2 + K^2)$.

## Appendix B Proof of Lemma 1

*Proof.* It suffices to inspect the projection of an individual differential. Notice that $\Pi_{\mathcal{D}^K}(C)$ boils down to the projection of the singular values $\{\sigma_i\}$ onto a convex cone. The proof follows by noticing that a nontrivial (Euclidean) projection onto a convex cone is always strictly contractive. This can be seen by considering the right angled triangle whose vertices are the differential $C$, its projection and the origin, as illustrated in the inset. □




## Appendix C Proof of Lemma 2

*Proof.* Suppose that $\langle \mathbf{n}_0, \boldsymbol{\xi} \rangle > 0$ for some one-sided tangent direction $\boldsymbol{\xi}$, and let $\gamma$ be the corresponding curve. Using the first order approximation of $\gamma$ and the definition of $\mathbf{n}_0$ we have,

$$\begin{aligned} \|\mathbf{z}_0 - \gamma(t)\|_2^2 &= \left\|\mathbf{z}_0 - \Pi_{\mathcal{D}}(\mathbf{z}_0) - t\boldsymbol{\xi} + O(t^2)\right\|_2^2 \\ &= \left\|\mathbf{n}_0 - t\boldsymbol{\xi} + O(t^2)\right\|_2^2 \\ &= \|\mathbf{n}_0\|_2^2 - 2t\,\langle \mathbf{n}_0, \boldsymbol{\xi} \rangle + O(t^2). \end{aligned}$$

Since $\langle \mathbf{n}_0, \boldsymbol{\xi} \rangle > 0$, for sufficiently small $t > 0$ we have

$$\|\mathbf{z}_0 - \gamma(t)\|_2^2 < \|\mathbf{n}_0\|_2^2 = \|\mathbf{z}_0 - \Pi_{\mathcal{D}}(\mathbf{z}_0)\|_2^2,$$

in contradiction to $\Pi_{\mathcal{D}}(\mathbf{z}_0)$ being the projection of $\mathbf{z}_0$ onto $\mathcal{D}$. □

## Appendix D Efficient linear step using pre-factorization

Following is the derivation of the solution (10) for the linear KKT system (8). Solving (9) for the the pair $(\mathbf{x}, \boldsymbol{\lambda})$ gives

$$\begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = F_M(\mathbf{c}_0 - \mu\boldsymbol{\eta}_0) = F_M(\mathbf{c}_0) - \mu F_M(\boldsymbol{\eta}_0). \tag{17}$$

Substituting into the last equation of (9) yields

$$\boldsymbol{\eta}_0{}^T \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \boldsymbol{\eta}_0{}^T F_M(\mathbf{c}_0) - \mu \boldsymbol{\eta}_0{}^T F_M(\boldsymbol{\eta}_0) = d_0.$$

Isolating $\mu$ results with

$$\mu = \frac{\boldsymbol{\eta}_0{}^T F_M(\mathbf{c}_0) - d_0}{\boldsymbol{\eta}_0{}^T F_M(\boldsymbol{\eta}_0)}.$$

Substituting $\mu$ into (17) concludes the derivation.