Eurographics Symposium on Geometry Processing 2012
Eitan Grinspun and Niloy Mitra
(Guest Editors)

Volume 31 (2012), Number 5

Shape-Up: Shaping Discrete Geometry with Projections

Sofien Bouaziz   Mario Deuss   Yuliy Schwartzburg   Thibaut Weise   Mark Pauly

École Polytechnique Fédérale de Lausanne, Switzerland

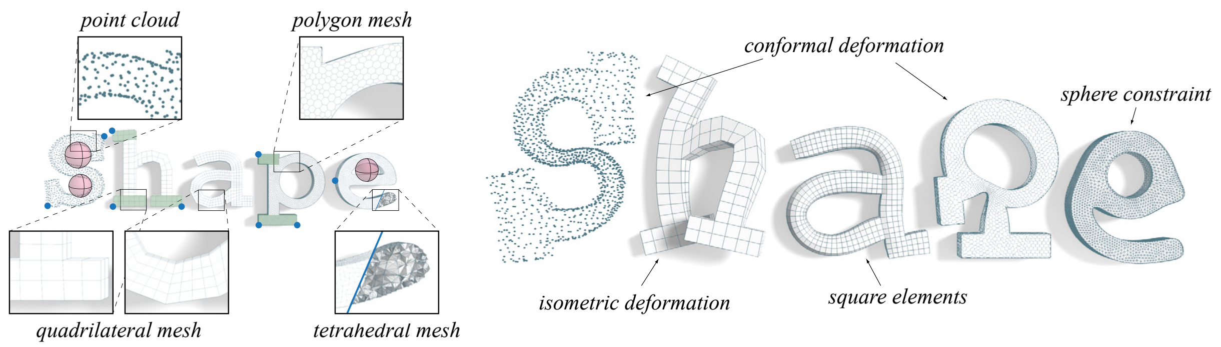


Figure 1: *Constraint-based optimization and interactive shape exploration on different geometry representations. Blue dots denote handle positions, green areas are constrained to remain rigid and red spheres indicate that vertices should be arranged on a sphere.*

Abstract

We introduce a unified optimization framework for geometry processing based on shape constraints. These constraints preserve or prescribe the shape of subsets of the points of a geometric data set, such as polygons, one-ring cells, volume elements, or feature curves. Our method is based on two key concepts: a shape proximity function *and* shape projection operators*. The proximity function encodes the distance of a desired least-squares fitted elementary target shape to the corresponding vertices of the 3D model. Projection operators are employed to minimize the proximity function by relocating vertices in a minimal way to match the imposed shape constraints. We demonstrate that this approach leads to a simple, robust, and efficient algorithm that allows implementing a variety of geometry processing applications, simply by combining suitable projection operators. We show examples for computing planar and circular meshes, shape space exploration, mesh quality improvement, shape-preserving deformation, and conformal parametrization. Our optimization framework provides a systematic way of building new solvers for geometry processing and produces similar or better results than state-of-the-art methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems;

1. Introduction

Geometry processing is commonly concerned with editing, optimizing, or otherwise transforming geometric data. An important goal in many applications is to obtain or preserve certain geometric shapes within this data. By shapes we mean the spatial relation of subsets of points in the geometry, such as mesh polygons, one-ring neighborhoods, curves in a mesh, etc. For example, equilateral triangles or tetrahedra are often desirable in surface or volume meshing. Meshes with planar polygons, or polygons whose vertices all lie on a circle, are of great interest in architectural geometry, since they directly relate to benefits in physical production. In interactive design, lengths or angles should commonly be preserved as much as possible when deforming a 3D model, leading to near-isometric and quasi-conformal deformation

NB

methods that favor rigid motions or similarity transforms of each of the mesh elements. Similarly, preserving element shapes is essential when computing a 2D parameterization of a 3D model.

All these application examples share the common trait that they transform a polygonal mesh such that sets of vertices preserve or assume predefined shapes. We show that this class of geometric problems can be formulated in a unified framework, leading to simpler, more robust, and in some instances more efficient implementations than current state-of-the-art algorithms.

We introduce two key concepts: a *shape proximity function* and *shape projection operators*. The proximity function encodes the distance of a desired least-squares fitted elementary target shape, e.g. a regular polygon, to the corresponding vertices in the mesh. Our optimization method uses projection operators as the main ingredient to minimize the proximity function. These operators relocate vertices in a minimal way to match the imposed shape constraints. We show that our formulation has several important advantages:

- **Unification.** The general problem of prescribing shapes on discrete geometric data sets can be solved in a unified manner, i.e. with one optimization framework by simply combining suitable projection operators. Our approach is not restricted to triangle or quadrilateral meshes, but is applicable to meshes with arbitrary degree polygons, volume meshes, point clouds, or other discrete geometry representations (see Figure 1).
- **Robustness.** Numerous algorithms rely on the derivative of angles with respect to the vertex positions or need to divide by edge length or face area. These computations become numerically unstable as soon as an element degenerates, leading to a premature halt of the optimization. Our solution is based on least-squares shape matching, resulting in robust numerics even in the presence of degenerate elements.
- **Simplicity.** Our framework allows solving geometry processing problems by simply defining the least-squares fit of a desired shape to a set of vertices. This conceptual simplicity translates directly into simplicity of implementation, which is crucial for integration into existing systems or adaptation to new geometric problems and application domains.

Related Work. Enforcing shape constraints by least-squares fitting has been used successfully for interactive editing tools and physics-based simulation [IMH05, MHTG05, BPGK06, GSMCO09]. Our approach is most closely related to the as-rigid-as-possible deformation method of Sorkine and Alexa [SA07] in that we also employ a two-step optimization strategy for shape constraint optimization. In this paper we unify, formalize, and extend this concept, and show how it can be applied to solve a large variety of different problems in geometry processing.

We review some of these applications that until now have typically been solved by specialized optimization algorithms tailored to certain application domains. This review does not aim for completeness, but rather provides an overview of the scope of algorithms that our framework encompasses.

Deformation and parametrization are two prominent applications where preservation of certain geometric features is crucial, such as the shape of polygons or one-ring neighborhoods. Near-isometric and quasi-conformal methods for deformations [IMH05, BPGK06, SA07, EP09, SBCBG11, CPS11] or parametrization [LPRM02, DMA02, LZX*08, MTAD08, AW11] aim to preserve lengths or angles, respectively. More recently, mesh editing tools have been developed [MYF07, GSMCO09] that integrate shape constraints on larger compound structures such as feature curves, allowing for intuitive deformations with high-level feature preservation.

For many geometry processing algorithms and especially finite element analysis, the accuracy of computation depends on the size and shape of the elements. Numerous methods have been designed to improve numerics by relocating vertices on the mesh in order to minimize certain error measures based on element size, shape, and smoothness [Mun07, ZBX09]. These methods are often combined with remeshing algorithms [BK04, TACSD06] in order to further ~~improve the elements~~ of a mesh.

NB

In freeform architecture, mechanical engineering, and product design, curves and surfaces are commonly split into pieces that can be manufactured separately. Optimizing the shape of those pieces is important to facilitate production or reduce manufacturing cost. Several rationalization methods have been proposed that optimize vertex positions in a mesh to satisfy certain geometric properties imposed by physical production. Planar, conical, or circular meshes [LPW*06, CW07, ZSW10, LXW*11], and planar, circular, or geodesic curves [DPW11] are examples of such shape optimization methods in architectural design.

Contributions. The core contribution of this paper is the unification of shape constraints into a single energy formulation. We introduce a toolbox of projection operators that allows reformulating many of the above cited methods in one optimization framework that is simple, robust, and leads to efficient implementations. Beyond shedding new light on these existing methods by highlighting their similarities, our algorithm facilitates plug-and-play design of new optimization methods. Implementing and combining suitable projection operators provides a flexible tool for constraint-based shape optimization and exploration that is applicable in many different application domains.

2. Shaping Discrete Geometry

The main feature of our method is to prescribe shape constraints for sets of points in a geometric model. We first de- NB

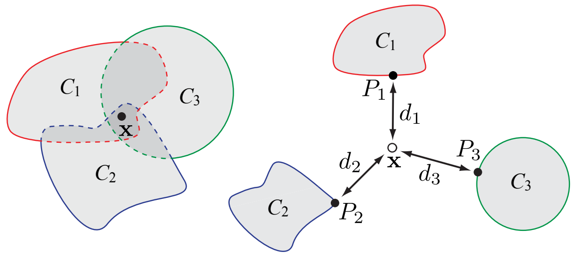


Figure 2: *The proximity function $\phi(\boldsymbol{x})$ is the weighted sum of squared distances $d_i(\boldsymbol{x})$ of the point $\boldsymbol{x}$ to the projections $P_i(\boldsymbol{x})$ onto the respective constraint sets C_i. Minimizing $\phi(\boldsymbol{x})$ yields a feasible solution if the constraint sets intersect (left), and a least-squares solution otherwise (right).*

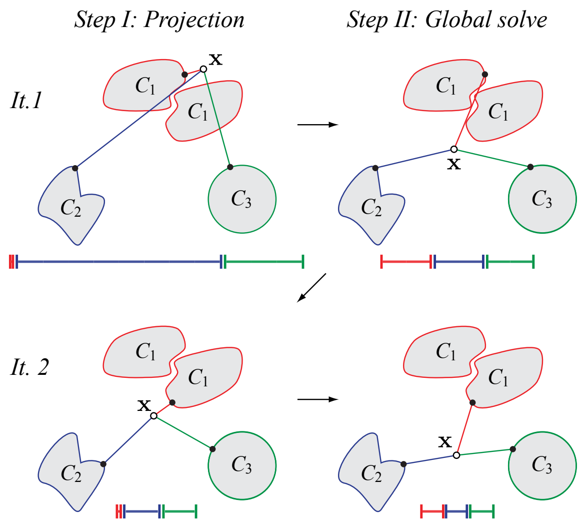


Figure 3: *Two iterations of the two-step minimization of the proximity function $\phi(\boldsymbol{x})$ with $w_i = 1$. Step I computes the projections using the current estimate $\boldsymbol{x}$. Step II updates $\boldsymbol{x}$ by minimizing $\phi(\boldsymbol{x})$ keeping the projections fixed. At each step, $\phi(\boldsymbol{x})$, illustrated by the sum of the error bars, will decrease, even if some of the individual elements increase.*

scribe the general approach for constraint satisfaction based on projection, then adapt this algorithm to the domain of discrete geometry.

2.1. Proximity Function

We draw inspiration from a technique applied in the signal processing community for constraint satisfaction problems that may not have feasible solutions [Com94]. Central to the method is a *proximity function* that measures the weighted sum of squared distances of a point to a collection of constraint sets, i.e. the sets containing feasible solutions to their respective constraints. For a collection of constraint sets $\{C_1, C_2, ..., C_m\}$, let $d_i(\mathbf{x})$ measure the 'least amount of change' in $\mathbf{x} \in \mathbb{R}^n$ in order to satisfy the constraint C_i. The proximity function is then defined as

$$\phi(\mathbf{x}) = \sum_{i=1}^{m} w_i d_i(\mathbf{x})^2, \tag{1}$$

where w_i are non-negative weights that control the relative importance of the different constraints. Formally, d_i is the distance between a point $\mathbf{x}$ and its projection $P_i(\mathbf{x})$ onto the constraint set C_i (see Figure 2). We can formulate this projection as

$$\mathbf{y} = P_i(\mathbf{x}) = \underset{\mathbf{y} \in C_i}{\arg\min} \, ||\mathbf{y} - \mathbf{x}||_2^2, \tag{2}$$

which can be seen as moving $\mathbf{x}$ in the minimal way to satisfy the constraint. The proximity function can now be written as

$$\phi(\mathbf{x}) = \sum_{i=1}^{m} w_i ||\mathbf{x} - P_i(\mathbf{x})||_2^2. \tag{3}$$

This function encodes how well the constraints are satisfied through a distance measure. Finding a solution that minimizes the proximity function will therefore satisfy all the constraints if $\phi(\mathbf{x}) = 0$. Otherwise, a least-squares solution is obtained.

For linear projections P_i the global optimum is found using standard linear least-squares. Often, however, the projections are nonlinear and do not have an intuitive gradient. We therefore employ an iterative two-step minimization strategy:

I Compute the projections $P_i(\mathbf{x})$ using the current estimate $\mathbf{x}$.

II Update $\mathbf{x}$ by minimizing Equation 3, keeping $P_i(\mathbf{x})$ fixed.

This scheme is guaranteed to converge monotonically to a local minimum, even though this minimum is not necessarily reached in a finite number of steps. The convergence rate depends on the conditions of the problem and the projection functions involved. To understand why the optimization converges, we observe that step I weakly decreases each constraint cost $||\mathbf{x} - P_i(\mathbf{x})||_2^2$ given the current estimate $\mathbf{x}$, hence $\phi(\mathbf{x})$ cannot increase. Step II minimizes Equation 3 globally for a fixed $P_i(\mathbf{x})$, thus $\phi(\mathbf{x})$ also cannot increase. As a consequence, we obtain a sequence that is non-increasing and bounded from below (as mean-square errors cannot be negative), a sufficient condition for convergence to a local minimum. This argumentation is similar in spirit to the convergence proof exposed in [BM92] for the Iterative Closest Point (ICP) algorithm. The two step process is illustrated in Figure 3.

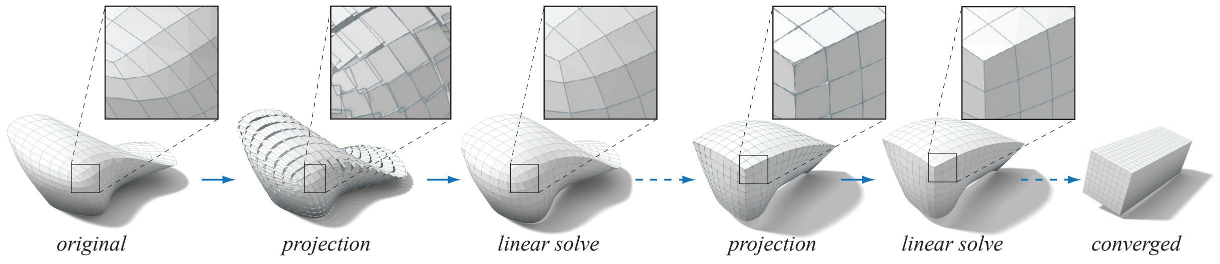


Figure 4: *Our optimization alternates between projection and linear solve. In this example, we prescribe a regular polygon constraint that pushes all quadrilaterals to become squares. The projection finds the best matching square for each quadrilateral to determine the target position for each vertex. The linear solve reconciles these projected positions in a least-squares sense.*

2.2. Shape Proximity for Geometric Data

The key observation of this paper is that the proximity function is ideally suited to encode geometric shape constraints. The projection of a set of vertices onto a geometric shape is found by minimizing the sum of the squared distances of the vertices to the corresponding constraint set. This minimum is computed through shape matching, i.e. by finding the least-squares fit of the constraint shape onto the set of vertices. Let $\mathbf{V}$ be a vector that stacks all vertices $\mathbf{v}_1,\ldots,\mathbf{v}_n \in \mathbb{R}^d$ of our d-dimensional data set and let $\mathbf{V}_i \subseteq \mathbf{V}$ be the n_i vertices involved in shape constraint C_i. We formulate the shape proximity function as

$$\phi(\mathbf{V}) = \sum_{i=1}^{m} w_i ||\mathbf{N}_i\mathbf{V}_i - P_i(\mathbf{N}_i\mathbf{V}_i)||_2^2, \qquad (4)$$

where w_i are weights and $P_i(\cdot)$ is the projection onto the constraint C_i, i.e. the corresponding least-squares fitted shape. The matrix $\mathbf{N}_i$ is used to center the vertices of $\mathbf{V}_i$ at their mean and is defined as

$$\mathbf{N}_i = (\mathbf{I}_{n_i\times n_i} - \frac{1}{n_i}\mathbf{1}_{n_i\times n_i}) \otimes \mathbf{I}_{d\times d}, \qquad (5)$$

where $\otimes$ is the Kronecker product and $\mathbf{1}_{n_i\times n_i}$ is a $n_i \times n_i$ matrix of ones. Subtracting the mean allows translational motion as a degree of freedom during the optimization. This introduces a global solve, but considerably improves convergence (see also Figure 10). This formulation is possible because shape projections are invariant under translation. Equation 4 can be reformulated by rewriting $\phi(\mathbf{V})$ as

$$E_{\text{shape}} = \phi(\mathbf{V}) = ||\mathbf{QV} - \mathbf{p}||_2^2, \qquad (6)$$

where the matrix $\mathbf{Q}$ combines all weighted mean-centered constraint vertices, and $\mathbf{p}$ integrates all projections. The alternating optimization scheme for each iteration then becomes:

I For fixed $\mathbf{V}$, compute the projection vector $\mathbf{p}$ using shape matching.

II For fixed $\mathbf{p}$, solve the normal equations $\mathbf{Q}^T\mathbf{QV} = \mathbf{Q}^T\mathbf{p}$ to update $\mathbf{V}$.

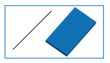
Since $\mathbf{Q}$ only depends on the shape constraints, we can pre-factor the matrix $\mathbf{Q}^T\mathbf{Q}$ using sparse Cholesky factorization. Figure 4 illustrates our two-step optimization scheme. In the projection step, we first compute the best fitting shape for each shape constraint. From the fitted shapes, we obtain the projected vertex positions and solve the linear system by back substitution using the prefactored matrix.

3. Shape Constraints and Projections

The core ingredient of our optimization framework are the shape projection operators. As mentioned above, we find the minimal displacement of vertices by projecting them onto the least-squares fit of the shape over those vertices. In this section we present a variety of different shape projections that can be combined, adapted, or extended to formulate new geometric optimization solutions. To simplify notation, we now denote with $\mathbf{V} = \{\mathbf{v}_1,\ldots,\mathbf{v}_n\}$ the vertices of a single constraint C_i (and not the full dataset) in the current configuration, and assume that these vertices are already mean centered. The original vertex positions are denoted by an apostrophe, i.e. $\mathbf{V}' = \{\mathbf{v}'_1,\ldots,\mathbf{v}'_n\}$, and the projected vertex positions by a star, i.e. $\mathbf{V}^* = \{\mathbf{v}^*_1,\ldots,\mathbf{v}^*_n\}$.

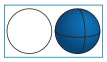
We describe three classes of constraints. *Continuous shapes*, such as planes or circles, *polygonal shapes*, such as line segments, regular polygons, or rectangles, and *relative shapes*. The latter encode the class of transformations that the shapes of the original geometry, e.g. polygons, tetrahedra, one-ring neighborhoods, etc., can undergo during the optimization. This allows the preservation of geometric properties such as lengths or angles of the original model.

3.1. Continuous Shapes


Line - Plane. This constraint specifies that the vertices of $\mathbf{V}$ should all lie on a continuous line or plane.

Projection: We can efficiently solve for the projection by first computing the sorted eigenvectors $\mathbf{U} = \left[\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\right]$ of the 3×3 covariance matrix $\mathbf{C}^T\mathbf{C}$ where $\mathbf{C} = \left[\mathbf{v}_1,\ldots,\mathbf{v}_n\right]$.

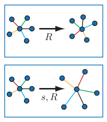
We remove the last column of $\mathbf{U}$ for plane projection and the last two columns for line projection. The projected vertices are then given as $\left[\mathbf{v}_1^*, ..., \mathbf{v}_n^*\right] = \mathbf{U}\mathbf{U}^T\mathbf{C}$.



Circle - Sphere. This constraint specifies that the vertices of $\mathbf{V}$ should all lie on a 2D circle or a 3D sphere.

Projection: Since the direct projection of 3D vertices to their 2D least-squares circle can be computationally expensive, we apply an approximate projection. We first project the vertices onto their least-squares plane (see above) and then fit a 2D circle within that plane. Circle fitting is achieved by minimizing $\sum_j (||\mathbf{v}_j - \mathbf{c}||_2^2 - r^2)^2$, where r and $\mathbf{c}$ are the unknown radius and center of the circle, respectively. We solve for these parameters using the closed-form solution of [TC89] and project the vertices of $\mathbf{V}$ onto this circle to obtain $\mathbf{V}^*$. The projection onto a sphere is computed by minimizing the same equation directly on the 3D points.

3.2. Relative Shapes



Rigid - Similar. These constraints are defined relative to the original vertex set $\mathbf{V}'$, i.e. they constrain the type of transformation that the vertex set can undergo. *Rigid* aims at restricting the deformations to isometries, while *Similar* aims for a conformal deformation.

Projection: Finding the closest rigid transform or similarity that maps the original vertices $\mathbf{V}'$ onto the current set $\mathbf{V}$ can be solved using the method described in [Ume91]. The algorithm computes the rigid transformation and uniform scale using least-squares fitting and allows a minimal and maximal scale constraint by keeping the rigid transformation as is and clamping the scale to the desired range.

While this approach works well, we also propose a faster projection operator for 2D shapes. The idea is to first project the vertices onto their least-squares plane and then formulate the fitting in 2D. We denote the projected 2D points by a bar, e.g. $\overline{\mathbf{v}}'_j$ is the projection of the original vertex $\mathbf{v}'_j$ onto the least-squares plane.

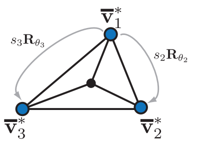
Let $\mathbf{M}$ be all the sets of points conformal to the 2D points $\overline{\mathbf{V}}' = \{\overline{\mathbf{v}}'_1, \ldots, \overline{\mathbf{v}}'_n\}$. We first find the point set $\overline{\mathbf{V}}^* \in \mathbf{M}$ closest to $\overline{\mathbf{V}}$, i.e. solve for

$$\{\overline{\mathbf{v}}_1^*, \ldots, \overline{\mathbf{v}}_n^*\} = \underset{\overline{\mathbf{V}}^* \in \mathbf{M}}{\arg\min} \sum_{j=1}^{n} ||\overline{\mathbf{v}}_j^* - \overline{\mathbf{v}}_j||_2^2. \tag{7}$$

As explained in [Hor87], at the minimum of Equation 7 the centroids of $\overline{\mathbf{V}}$ and $\overline{\mathbf{V}}^*$ coincide. Therefore, if $\overline{\mathbf{V}}$ is centered, Equation 7 can be expressed as

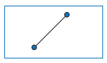
$$\underset{\overline{\mathbf{v}}_{1x}^*, \overline{\mathbf{v}}_{1y}^*}{\arg\min} \, || \underbrace{\begin{bmatrix} \mathbf{I}_{2\times 2} \\ s_2\mathbf{R}_{\theta_2} \\ \vdots \\ s_n\mathbf{R}_{\theta_n} \end{bmatrix}}_{\mathbf{A}} \underbrace{\overline{\mathbf{v}}_1^*}_{\mathbf{x}} - \underbrace{\begin{bmatrix} \overline{\mathbf{v}}_1 \\ \overline{\mathbf{v}}_2 \\ \vdots \\ \overline{\mathbf{v}}_n \end{bmatrix}}_{\mathbf{b}} ||_2^2, \tag{8}$$

where $s_i\mathbf{R}_{\theta_i}$ represent the scale and rotation mapping the first point to the ith point in the original centered set $\overline{\mathbf{V}}'$.


$s_3\mathbf{R}_{\theta_3}$
$\overline{\mathbf{v}}_1^*$
$s_2\mathbf{R}_{\theta_2}$
$\overline{\mathbf{v}}_3^*$
$\overline{\mathbf{v}}_2^*$

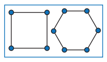
The minimum $\mathbf{x}$ of Equation 8 is obtained by solving the normal equation $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$. We can then express the projection as a linear operator $\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, which maps the current point set $\overline{\mathbf{V}}$ to the closest point set $\overline{\mathbf{V}}^*$ in $\mathbf{M}$. The matrix $\mathbf{P}$ depends only on the original point set $\overline{\mathbf{V}}'$ and can thus be precomputed. If $\mathbf{P}$ is applied to any point set in $\mathbf{M}$, by the idempotence property of the projection operator, the result is unchanged. Since $\mathbf{A}^T\mathbf{A}$ is a 2×2 matrix, this projection operator has a closed form expression.

3.3. Polygonal Shapes



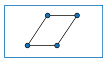
Line Segment. For a pair of vertices $\{\mathbf{v}_1, \mathbf{v}_2\}$, this constraint specifies the allowed value for their relative distance.

Projection: Let $d = ||\mathbf{v}_1 - \mathbf{v}_2||_2$ be the current distance between the vertices and d^* the desired length of the line segment. Then the projection $\{\mathbf{v}_1^*, \mathbf{v}_2^*\}$ is computed as $\mathbf{v}_1^* = \frac{d^*}{d}\mathbf{v}_1$ and $\mathbf{v}_2^* = -\mathbf{v}_1^*$.

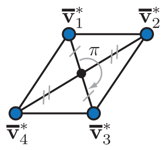


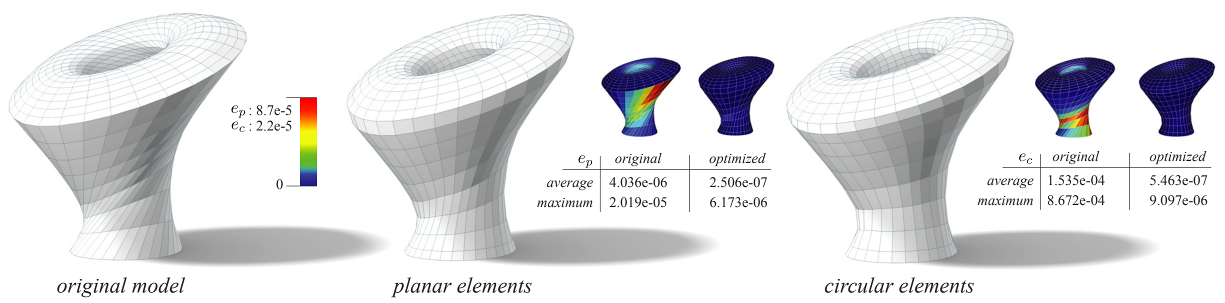
Regular Polygon. This constraint specifies that the vertex set $\mathbf{V}$ should assume the shape of a regular polygon, i.e. have all angles be equal and all sides be of equal length.

Projection: Since a regular polygon is invariant only under similarity transformations, we can use the same projection method as described above for *relative shapes*. We simply replace the original vertex set $\mathbf{V}'$ by the vertices of the regular polygon of the corresponding order.



Parallelogram. This constraint specifies that a quadrilateral should become a parallelogram, i.e. have two pairs of parallel sides.


$\overline{\mathbf{v}}_1^*$
$\overline{\mathbf{v}}_2^*$
π
$\overline{\mathbf{v}}_4^*$
$\overline{\mathbf{v}}_3^*$



e_p	original	optimized
average	4.036e-06	2.506e-07
maximum	2.019e-05	6.173e-06

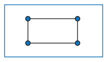
e_c	original	optimized
average	1.535e-04	5.463e-07
maximum	8.672e-04	9.097e-06

Figure 5: *An architectural design (left) optimized for planar (middle) and circular mesh elements (right). The colored images provide a visual comparison of the planarity error e_p and circularity error e_c. The error per face is the average squared distance of its vertices to the least-squares fit. In addition to shape constraints, we apply closeness and smoothness terms, using weights $(\lambda_{shape}, \lambda_{close}, \lambda_{smooth}) = (5, 10, 2)$ in Equation 13. The bounding sphere diameter of the object is 1.*

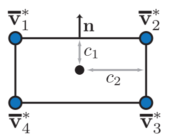
Projection: We formulate the parallelogram fitting by extending the projection for *relative shapes* as described above. We first project the vertices onto their least-squares plane, then formulate the optimization as

$$\arg\min_{\bar{\mathbf{v}}_1^*, \bar{\mathbf{v}}_2^*} || \underbrace{\begin{bmatrix} \mathbf{I}_{4\times4} \\ -\mathbf{I}_{4\times4} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \bar{\mathbf{v}}_1^* \\ \bar{\mathbf{v}}_2^* \end{bmatrix}}_{\mathbf{x}} - \underbrace{\begin{bmatrix} \bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_2 \\ \bar{\mathbf{v}}_3 \\ \bar{\mathbf{v}}_4 \end{bmatrix}}_{\mathbf{b}} ||_2^2. \tag{9}$$

As previously, the solution of this optimization is $\mathbf{V}^* = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T b$.



Rectangle. This constraint specifies that a quadrilateral should become a rectangle, i.e. have only right angles.



Projection: We first project the vertices onto their least-squares plane and then fit the rectangle in 2D. Unlike the other polygonal shapes, we compute the equation of the four lines that define the rectangle by solving

$$\arg\min_{c_1, c_2, \mathbf{n}} || \underbrace{\begin{bmatrix} 1 & 0 & \bar{\mathbf{v}}_{1x} & \bar{\mathbf{v}}_{1y} \\ 1 & 0 & \bar{\mathbf{v}}_{2x} & \bar{\mathbf{v}}_{2y} \\ 0 & 1 & \bar{\mathbf{v}}_{2y} & -\bar{\mathbf{v}}_{2x} \\ 0 & 1 & \bar{\mathbf{v}}_{3y} & -\bar{\mathbf{v}}_{3x} \\ -1 & 0 & \bar{\mathbf{v}}_{3x} & \bar{\mathbf{v}}_{3y} \\ -1 & 0 & \bar{\mathbf{v}}_{4x} & \bar{\mathbf{v}}_{4y} \\ 0 & -1 & \bar{\mathbf{v}}_{4y} & -\bar{\mathbf{v}}_{4x} \\ 0 & -1 & \bar{\mathbf{v}}_{1y} & -\bar{\mathbf{v}}_{1x} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \mathbf{n}_x \\ \mathbf{n}_y \end{bmatrix}}_{\mathbf{x}} ||_2^2 \quad \text{s.t} \quad ||\mathbf{n}||_2^2 = 1. \tag{10}$$

This optimization is minimized by taking the QR decomposition of $\mathbf{A}$ and solving a 2×2 eigenvalue problem as described in [GH95]. We then find the projected points by computing the intersection of these four lines.

As we show below, the projection operators introduced here provide a versatile toolbox for constructing geometric optimization methods. Other constraints, e.g. projection onto a spline curve, a developable surface, or explicit surface geometry, offer numerous opportunities for extending our framework and designing new projection-based solvers.

4. Applications

Before we evaluate the behavior and performance of our shape optimization framework, we highlight several applications. Beyond the shape constraints expressed in the proximity function, these applications typically have other objectives that can be directly integrated into our approach by defining suitable energy functions. In our examples, we use two such additional energies, a smoothness term and an energy that penalizes deviation from a given reference surface.

The closeness energy measures the distance of a vertex $\mathbf{v}_i$ to the original surface as

$$E_{\text{close}} = \sum_{i=1}^{n} ||\mathbf{v}_i - \mathbf{c}(\mathbf{v}_i)||_2^2, \tag{11}$$

where $\mathbf{c}(\mathbf{v}_i)$ is the closest point on the original surface to the vertex $\mathbf{v}_i$. We use a similar energy for boundary preservation and handle-based deformation. For smoothing, we use a Laplacian energy [BKP*10] of the form

$$E_{\text{smooth}} = \sum_{i=1}^{n} || \sum_{\{i,j\} \in \mathbf{E}} w_{ij}(\mathbf{l}_j - \mathbf{l}_i) ||_2^2, \tag{12}$$

where $\mathbf{E}$ denotes the mesh edges, $\mathbf{l}_i = \mathbf{v}_i$ for the surface smoothing energy and $\mathbf{l}_i = \mathbf{v}_i - \mathbf{v}_i'$ for smoothing the deformation. We set the scalars w_{ij} to the standard cotangent weights for triangular meshes and uniform weights

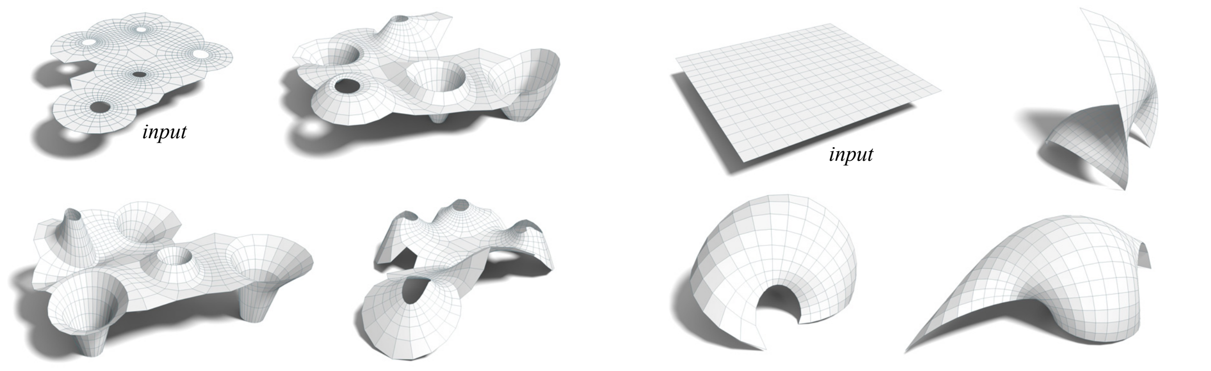


Figure 6: *Complex shape spaces can be defined by combining projection operators. Left: circle constraints on the polygons maintain the circular property of the mesh during interactive shape exploration. Right: circle constraints on the straight line curves of the input mesh define a functional web. A similarity constraint on the mesh elements leads to a quasi-conformal deformation. No smoothing energy or closeness term is used in Equation 13, except to define positional constraints for the deformation handles (see video).*

otherwise. Alternatively, the weighting scheme proposed in [AW11] could be used for general polygonal meshes.

The final energy used in our framework is a convex combination of the previous energies

$$E_{\text{final}} = \lambda_{\text{shape}} E_{\text{shape}} + \lambda_{\text{close}} E_{\text{close}} + \lambda_{\text{smooth}} E_{\text{smooth}}, \quad (13)$$

where the weights $(\lambda_{\text{shape}}, \lambda_{\text{close}}, \lambda_{\text{smooth}})$ are parameters that can be selected by the user. By reformulating E_{final} analogously to Equation 6, we can minimize the energy with our alternating projection method (see also Figure 4). This general procedure can be applied to numerous different domains using exactly the same optimization code, simply by combining suitable projection operators. We demonstrate this by illustrating applications in architectural geometry, mesh quality improvement, interactive deformation, and parameterization. Note that even when adding the additional energies for closeness and smoothness, the convergence guarantee of Section 2.1 remains valid. In step I of the optimization, vertices are kept fixed, so the projection has no influence on E_{close} and E_{smooth}, and hence E_{final} decreases weakly. Step II is a global minimization of Equation 13 over the vertex positions, so E_{final} cannot increase either.

4.1. Planar and Circular Constraints

Planar and circular meshes, i.e. meshes in which the vertices of each polygonal face lie on a plane or circle, are important in the field of architectural design, since they directly relate to benefits in physical production [PW08]. We can optimize for element planarity and circularity using the plane and circle projections introduced above. Contrary to previous work [LPW*06, ZSW10, LXW*11] our approach is not restricted to quadrilateral meshes, since the projections are defined for arbitrary vertex sets. We can even apply the same solver to optimize for planar or circular curves on a surface to generate functional webs similar to [DPW11]. Figure 5 shows examples of architectural models that have been optimized with our approach. We also provide a comparison with the method of [LPW*06] in Figure 11, illustrating the benefits of our approach.

Planar and circular constraints can also be combined with relative shape constraints (see Sections 3.2 and 4.3) to enable interactive exploration of the space of planar or circular meshes with fixed connectivity, similar to [YYPM11]. An example of shape exploration using a handle-based editing metaphor is illustrated in Figure 6. Compared to [YYPM11],

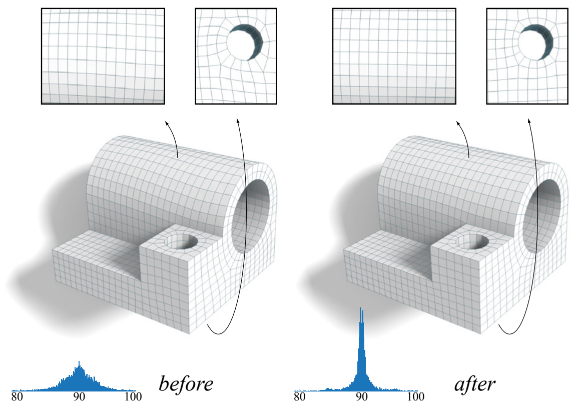


Figure 7: *Mesh quality optimization. The quadrilateral surface on the left has been generated with the meshing algorithm of [BZK09]. Applying the square shape constraint to each element improves the angle distribution as illustrated by the histograms. Weights for Equation 13:* $(\lambda_{shape}, \lambda_{close}, \lambda_{smooth}) = (5, 1, 0)$.

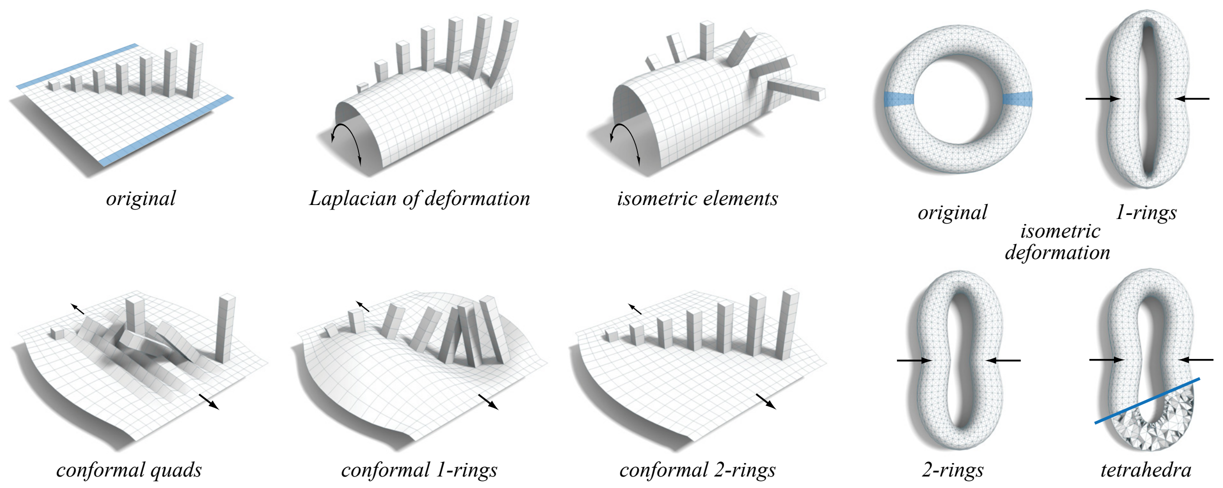


Figure 8: *A variety of shape preserving editing tools can be implemented with our framework. Top center: rigid constraints on the polygons lead to intuitive rotations of the protrusions. Bottom left: when stretching the plane with similarity constraints, different deformation behavior is obtained depending on the size and overlap of the elements. Right: comparison of rigid constraints on surface and volume elements. These examples use no smoothing energy or closeness term in Eq. 13, except to define positional constraints for the deformation handles (blue) dragged in the direction of the arrows.*

our implementation is substantially simpler and allows for more flexible shape exploration. We can apply the circular constraints not only on the mesh polygons, but on arbitrary vertex sets, including curves embedded in the surface. As the figure and the accompanying video illustrate, this leads to interesting new design tools suitable for architectural form finding and fabrication-aware design.

4.2. Mesh Quality Improvement

In geometry processing and especially in finite element analysis, the accuracy of computation depends on the size and shape of the elements. In numerous cases, having isotropic elements such as equilateral triangles or squares leads to better numerics (see [She02] for more details). Various methods have been designed to enhance mesh quality by iterating between topological remeshing operations to improve mesh connectivity, and vertex relocation to improve element shapes [AMD02, BK04, TWAD09]. Our approach is ideally suited for the second step, since we can directly prescribe the desired element shapes using the polygonal shape projections of Section 3.3. In Figure 7 we illustrate how a quadrilateral mesh that has been generated with the mixed-integer quadrangulation method of [BZK09] is optimized for better element shapes using the square projection operator. Corners and feature curves are fixed in this example, but vertices are allowed to slide along the feature curves.

4.3. Interactive Deformation

Surface and volume deformation algorithms have become an integral component of interactive design tools. The main goal of these methods is local shape preservation to allow intuitive shape deformation based on simple handle constraints controlled by the user. In particular, local feature preservation can be achieved by only allowing shape elements to translate, rotate, and possibly scale uniformly, leading to as-rigid-as-possible [IMH05, BPGK06, SA07] or quasi-conformal [EP09, SBCBG11, CPS11] deformation methods.

Our approach provides a general recipe for creating shape preserving deformation methods as illustrated in Figure 8. Using the relative shape constraints of Section 3.2, we can design a multitude of near-isometric and near-conformal deformation methods by applying the projection operators to different local subsets of vertices, such as polygons, volume elements, local 1-rings, 2-rings, etc. These different combinations allow tailoring the deformation model to a desired behavior and specific application context.

Parametrization. As shown in Figure 9 we can also obtain a discrete free-boundary conformal parameterization of arbitrary degree polygon meshes by setting the projection $P_i(\mathbf{N}_i\mathbf{V}_i)$ to $\mathbf{P}_i\mathbf{N}_i\mathbf{V}_i$ where $\mathbf{P}_i$ is the linear 2D projection defined in Section 3.2:

$$\phi(\mathbf{V}) = \sum_{i=1}^{m} w_i||(\mathbf{N}_i - \mathbf{P}_i\mathbf{N}_i)\mathbf{V}_i||_2^2. \qquad (14)$$

This is similar in spirit to the method proposed by [LZX*08] where each 3D element is projected onto the 2D parameterization domain and globally optimized. In our formulation this reduces to a homogeneous linear least-squares problem that we solve using a sparse eigenvalue solver. In order to be independent of the meshing (as seen in Figure 9), the weights w_i can be set to the inverse of the area of the original poly-

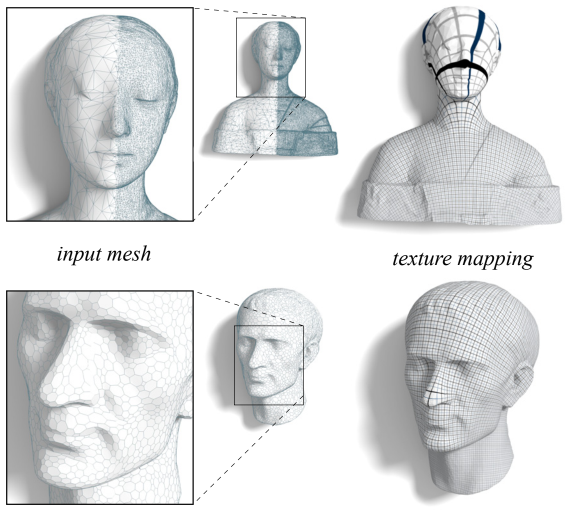


Figure 9: *A variant of similarity shape constraints are employed to compute free-boundary conformal parameterizations on an irregular triangle mesh (top) and a mesh with higher order polygons (bottom).*

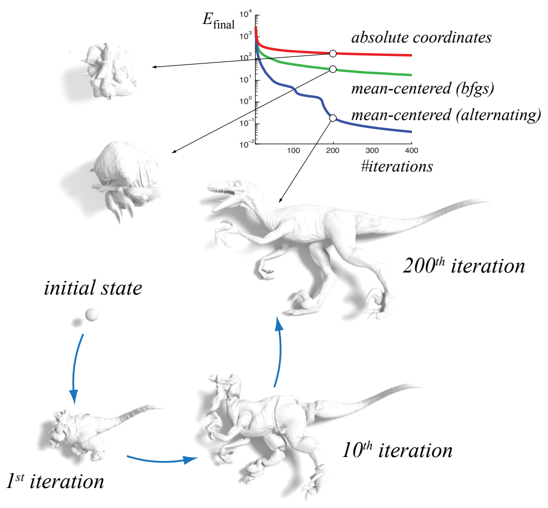


Figure 10: *Stability of our optimization. The algorithm is able to recover the original raptor model from a collapsed initial state by prescribing the original one-ring shapes. Centering the vertices at their mean for each shape constraint is essential for fast convergence as illustrated in the plots (red vs. blue). The alternating minimization performs significantly better than BFGS, a Quasi-Newton method (green vs. blue).*

gons as explained in [MTAD08]. Our parametrization for triangular meshes is visually similar to [MTAD08]. The difference in quasi-conformal error (as defined by [SSGH01]) is within ±0.004.

5. Discussion

In this section we analyze the behavior of the optimization, report performance data and implementation details, and discuss limitations of our approach.

Robustness. One important advantage of our approach is numerical stability. In Figure 10 we show how the optimization can recover from a highly corrupted initial configuration. Since our projection operators are stable even for degenerate vertex positions, we avoid the instabilities of many gradient-based methods that rely on derivates of angles with respect to vertices. Figure 11 illustrates how this improved robustness facilitates higher quality results than previous methods. In this example, the design objective of achieving smooth curves while keeping all elements circular is in conflict with the numerical requirement of avoiding collapsed edges on which previous work is dependent. With our method, we achieve higher overall smoothness by allowing degenerate elements that can easily be handled by the shape matching operators.

Performance. The gradient $\nabla\phi(\mathbf{x}) = 2\sum_{i=1}^{m} w_i(\mathbf{x} - P_i(\mathbf{x}))$ of the proximity function (Equation 3) is simple to compute, since it avoids the explicit computation of the partial derivatives of the projection function (see Appendix A). However, we found that the alternating optimization scheme discussed in Section 2.2 performs significantly better than simple gradient descent or BFGS, a Quasi-Newton method [NW00] (see convergence plots in Figure 10). Note that standard Gauss-Newton or Newton-type solvers are not easily applicable for this optimization problem, since they require the computation of the Jacobian or Hessian. In contrast to the gradient, these computations do require the evaluation of the partial derivatives of the projection function, which can be difficult to compute.

The number of iterations necessary for our solver to converge depends on the problem setting, i.e., the number of unknowns, number and type of constraints, and the specific convergence criteria. For parametrization, a model of about 35 thousand vertices is parametrized by our method in 3.5 seconds and performance scales approximately linearly with the number of vertices. For interactive shape exploration and deformation of medium sized models ($<$ 30K constraints and $<$ 30K unknowns), 10-20 iterations are usually sufficient when initializing the optimization with the previous frame. At 3-6ms per iteration, this enables interactive editing (10-30fps) on a Mac Pro 2 x 2.26GHz Quad-Core Intel Xeon with 16GB of memory (see also accompanying video). When the meshes grow in number of elements, more iterations are needed until convergence with a higher cost per iteration. The necessary performance improvements could be achieved by multi-resolution methods as demonstrated in

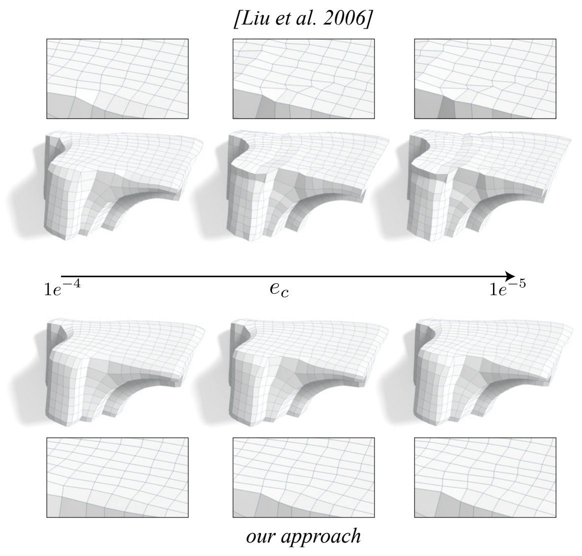


Figure 11: *Circular mesh optimization. Comparison of our method with [LPW*06] using the same fairness and smoothness weights at the same circular error e_c (sum of squared distances from the vertices to the fitted circle). Our method finds a better local minimum due to its numerical stability in the case of collapsing edges. The bounding sphere diameter of the object is 10.*

several previous works, e.g. [SYBF06, BPGK06]. We also plan to explore a GPU implementation as future work.

We have shown that most shape projections can be implemented efficiently using least-squares fitting. Certain shapes, however, can be difficult to fit, e.g. an ellipse, and may need an additional (non-linear) optimization step, which can be expensive. One alternative solution is to use fast, approximate least-squares fitting algorithms, as we have demonstrated with the circle fitting method of Section 3.1.

Implementation details. The complete framework presented in this paper is implemented in C++. We use the `eigs` function of MATLAB to solve the sparse eigenvalue problem for the parametrization. The Eigen library (eigen.tuxfamily.org) is employed for dense and sparse linear algebra computations and for the implementation of [Ume91]. The closest point search required for the computation of the closeness energy of Equation 11 is accelerated using a k-d tree. In our implementation we solve the optimization independently for each coordinate. We prefactorize $\mathbf{Q}^T\mathbf{Q}$ using sparse Cholesky factorization and perform three times back-substitution. Note that without any closeness term in E_{final} the matrix $\mathbf{Q}^T\mathbf{Q}$ is singular as it has translational degrees of freedom. Fixing one vertex is sufficient to remove this degeneracy.

Our algorithm is dependent on the choice of vertices involved in each constraint. Slightly different choices, i.e. exchanging two vertices between constraints, might lead to different results. The current implementation does not detect or avoid self-intersections and keeps a fixed mesh topology during the optimization. Certain applications might require dynamic modifications of vertex connectivity to achieve better results. In such cases our method could be combined with remeshing algorithms, which we plan to explore in the future.

6. Conclusion

We have presented a framework for processing discrete geometric data sets based on shape constraints. Our new formulation provides a simple, but effective recipe for geometry optimization suitable for numerous different application domains. Plug-and-play design of optimization methods becomes feasible by simply selecting or implementing suitable projection operators. Our experiments show that these solvers reproduce or outperform existing algorithms, often with the additional benefit of improved robustness and generalization to arbitrary degree polygons. Beyond the examples on mesh optimization, interactive deformation, and shape space exploration that we show, we believe that our framework can be applied to many other geometry processing tasks, including non-rigid registration [LAGP09], symmetrization [MGP07] and deformation transfer [SP04]. More importantly, it provides a simple and effective recipe to build new optimization solvers, thus enabling scientists and practitioners in different domains to easily integrate geometry processing tools into their applications.

NB

Acknowledgment. We thank Mario Botsch, Russell Luke, Thilo Rörig, and Alexander Schiftner for valuable discussions, Brian Amberg, Duygu Ceylan, Minh Dang, Bailin Deng, Laura Gosmino, Stefan Lienhard, and Juyong Zhang for proof-reading the paper and giving valuable comments, and Pierre Alliez, David Bommes, Daniel Piker, and AIM@SHAPE for providing 3D models. This research is supported by Swiss National Science Foundation grant 20PA21L_129607.

References

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. In *Proc. of ACM SIGGRAPH* (2002). 8

[AW11] ALEXA M., WARDETZKY M.: Discrete laplacians on general polygonal meshes. *ACM Trans. Graph. 30* (2011), 102:1–102:10. 2, 6

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *SGP* (2004). 2, 8

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. 6

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 14* (1992), 239–256. 3

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: Primo: coupled prisms for intuitive surface modeling. In *SGP* (2006). 2, 8, 9

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. In *ACM Trans. Graph.* (2009). 7, 8

[Com94] COMBETTES P.: Inconsistent signal feasibility problems: Least-squares solutions in a product space. *IEEE Transactions on Signal Processing 42* (1994), 2955–2966. 3

[CPS11] CRANE K., PINKALL U., SCHRÖDER P.: Spin transformations of discrete surfaces. *ACM Trans. Graph. 30* (2011), 104:1–104:10. 2, 8

[CW07] CUTLER B., WHITING E.: Constrained planar remeshing for architecture. In *Proc. of Graphics Interface* (2007). 2

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. *CGF 21* (2002), 209–218. 2

[DPW11] DENG B., POTTMANN H., WALLNER J.: Functional webs for freeform architecture. In *SGP* (2011). 2, 7

[EP09] EIGENSATZ M., PAULY M.: Positional, metric, and curvature control for constraint-based surface deformation. *Comput. Graph. Forum 28*, 2 (2009), 551–558. 2, 8

[GH95] GANDER W., HREBICEK J.: *Solving Problems in Scientific Computing Using Maple and MATLAB*. Springer-Verlag New York, 1995. 6

[GSMCO09] GAL R., SORKINE O., MITRA N., COHEN-OR D.: iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. 28* (2009), 33:1–33:10. 2

[Hor87] HORN B.: Closed-form solution of absolute orientation using unit quaternions. *J. of the Opt. Society of America A 4* (1987), 629–642. 5

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24* (2005), 1134–1141. 2, 8

[LAGP09] LI H., ADAMS B., GUIBAS L. J., PAULY M.: Robust single-view geometry and motion reconstruction. *ACM Trans. Graph. 28* (2009), 175:1–175:10. 10

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21* (2002), 362–371. 2

[LPW∗06] LIU Y., POTTMANN H., WALLNER J., YANG Y.-L., WANG W.: Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. 25* (2006), 681–689. 2, 7, 10

[LXW∗11] LIU Y., XU W., WANG J., ZHU L., GUO B., CHEN F., WANG G.: General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph. 30* (2011), 140:1–140:10. 2, 7

[LZX∗08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *SGP* (2008). 2, 9

[MGP07] MITRA N. J., GUIBAS L. J., PAULY M.: Symmetrization. *ACM Trans. Graph. 26* (2007). 10

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24* (2005), 471–478. 2

[MTAD08] MULLEN P., TONG Y., ALLIEZ P., DESBRUN M.: Spectral conformal parameterization. In *SGP* (2008). 2, 9

[Mun07] MUNSON T.: Mesh shape-quality optimization using the inverse mean-ratio metric. *Math. Programming 110* (2007), 561–590. 2

[MYF07] MASUDA H., YOSHIOKA Y., FURUKAWA Y.: Preserving form features in interactive mesh deformation. *Comput. Aided Des. 39* (2007), 361–368. 2

[NW00] NOCEDAL J., WRIGHT S. J.: *Numerical Optimization*. Springer, 2000. 9

[PW08] POTTMANN H., WALLNER J.: The focal geometry of circular and conical meshes. *Adv. Comp. Math 29* (2008), 249–268. 7

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *SGP* (2007). 2, 8

[SBCBG11] SOLOMON J., BEN-CHEN M., BUTSCHER A., GUIBAS L.: As-killing-as-possible vector fields for planar deformation. In *SGP* (2011). 2, 8

[She02] SHEWCHUK J. R.: What is a good linear element? - interpolation, conditioning, and quality measures. In *IMR* (2002). 8

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23* (2004), 399–405. 10

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *Proc. of ACM SIGGRAPH* (2001). 9

[SYBF06] SHI L., YU Y., BELL N., FENG W.-W.: A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. 25* (2006), 1108–1117. 9

[TACSD06] TONG Y., ALLIEZ P., COHEN-STEINER D., DESBRUN M.: Designing quadrangulations with discrete harmonic forms. In *SGP* (2006). 2

[TC89] THOMAS S., CHAN Y.: A simple approach for the estimation of circular arc center and its radius. *Computer Vision, Graphics, and Image Processing 45* (1989), 362–370. 5

[TWAD09] TOURNOIS J., WORMSER C., ALLIEZ P., DESBRUN M.: Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28* (2009), 75:1–75:9. 8

[Ume91] UMEYAMA S.: Least-squares estimation of transformation parameters between two point patterns. *PAMI 13* (1991), 376–380. 5, 10

[YYPM11] YANG Y.-L., YANG Y.-J., POTTMANN H., MITRA N. J.: Shape space exploration of constrained meshes. *ACM Trans. Graph. 30* (2011), 124:1–124:12. 7

[ZBX09] ZHANG Y., BAJAJ C., XU G.: Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Comm. in num. meth. in eng. 25* (2009), 1–18. 2

[ZSW10] ZADRAVEC M., SCHIFTNER A., WALLNER J.: Designing quad-dominant meshes with planar faces. *CGF 29* (2010), 1671–1679. 2, 7

Appendix A: Gradient of the Proximity Function

The gradient of the proximity function $\phi(\mathbf{x}) = ||\mathbf{x} - P(\mathbf{x})||_2^2$ can be computed as

$$\begin{aligned}\nabla\phi(\mathbf{x}) &= (\mathbf{I} - J(P(\mathbf{x})))^T 2(\mathbf{x} - P(\mathbf{x})) \\ &= 2(\mathbf{x} - P(\mathbf{x})) - J(P(\mathbf{x}))^T 2(\mathbf{x} - P(\mathbf{x})) \\ &= 2(\mathbf{x} - P(\mathbf{x})),\end{aligned}$$

where $J(P(\mathbf{x}))$ is the Jacobian of $P(\mathbf{x})$ and $\mathbf{I}$ is the identity matrix. For the partial derivatives of $P(\mathbf{x})$ (i.e. the columns of the Jacobian), the component in the direction of $(\mathbf{x} - P(\mathbf{x}))$ is zero, because $P(\mathbf{x})$ does not change when $\mathbf{x}$ is moving towards $P(\mathbf{x})$. The other components are orthogonal to $(\mathbf{x} - P(\mathbf{x}))$ and therefore $J(P(\mathbf{x}))^T(\mathbf{x} - P(\mathbf{x}))$ evaluates to 0.