A General Framework for Constrained Mesh Parameterization

Conference Paper · April 2015

DOI: 10.1145/2788539.2788541

2 authors:

Márton Vaitkus
Budapest University of Technology and Econo…
4 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Tamas Varady
Budapest University of Technology and Econo…
67 PUBLICATIONS **2,125** CITATIONS
SEE PROFILE

A General Framework for Constrained Mesh Parameterization

Márton Vaitkus*
Budapest University of Technology and Economics

Tamás Várady†
Budapest University of Technology and Economics

Abstract

Parameterizing or flattening a triangle mesh is necessary for many applications in computer graphics and geometry. While mesh parameterization is a very popular research topic, the vast majority of the literature is focused on minimizing distortion or satisfying constraints related to certain applications such as texturing or quadrilateral remeshing. Certain downstream applications require adherence to more general, geometric constraints – possibly at the cost of higher distortion. These geometric constraints include requirements such as certain vertices lie on some line or circle, or a planar curve or developable region keeps its shape during parameterization. We present a framework for enforcing such constraints, motivated by the As-Rigid-As-Possible parameterization method, and demonstrate its effectiveness through several examples.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

Keywords: mesh parameterization, geometric constraints, vector field, geometry processing

1 Introduction

The *parameterization*, or the *flattening* of 3D triangle meshes is a fundamental task in computer graphics and geometry with applications including, but not limited to: texturing, remeshing, surface fitting and mesh repair. The problem, due to its importance and inherent difficulty has been a topic of intense research, which continues to this day, as well.

In mesh parameterization our goal is to map a triangle mesh embedded in 3D to the plane, i.e. compute a pair of coordinates (usually denoted by u and v) for each point on the mesh. For meshes with disc topology, this mapping is continuous and piecewise-affine, which means that in practice it is sufficient to determine the coordinates of the mesh vertices. Restricted to a triangle, a parameterization is simply an affine mapping, which can be represented in local coordinates by a 2×2 matrix corresponding to the Jacobian of the function, see Figure 1. The distortion of the parameterization is quantified by considering these Jacobian matrices, which depend linearly on the vertex coordinates [Hormann et al. 2007].

Up until now, the research community has been mainly concerned with minimizing the distortion of the parameterization. A huge variety of distortion measures have been proposed along with efficient algorithms for their minimization. Nevertheless, there exist practical applications, where the demand to preserve and enforce certain geometric constraints is considered even more important than low geometric distortion. A large amount of work has concentrated on constraints involving only discrete vertex positions [Eckstein et al. 2001; Kraevoy et al. 2003] and constant coordinate lines [Bommes et al. 2009; Myles and Zorin 2013], mostly in the context of texture mapping and quadrilateral remeshing, respectively.

In this paper we consider a more general class of high-level parameterization constraints that define how certain entities on the mesh must be mapped to the domain. Some important examples of such *geometric constraints* are:

(C1) A sequence of edges is to be mapped to a line.

(C2) Vertices of a closed sequence of edges are to lie on a circle.

(C3) Angles between certain edges are prescribed, including orthogonality or parallelism.

(C4) A feature curve is to preserve its shape.

(C5) A planar or developable region is to preserve its shape.

To our knowledge – despite the vast amount of published research on parameterization – these constraints have not yet been considered in such generality; and we are aware of only a few isolated attempts to enforce certain proper subsets of them. Furthermore a common shortcoming of all previous approaches to constrained parameterization is that they ignore metric properties, i.e. they are able to influence the (relative) orientation of parts of the mesh, but unable to constrain their (relative) size. Our main contribution is a method to influence *simultaneously* the angles and edge lengths of the flattened mesh through constraints that are linear in the unknowns, within a two-phase parameterization algorithm similar to the As-Rigid-As-Possible and vector field-based global parameterization methods.

In what follows, we first give an overview of previous work and evaluate known parameterization algorithms according to their capability to enforce the constraints under consideration. Then, we describe a framework for enforcing geometric constraints during parameterization and demonstrate its effectiveness through several examples.

2 Previous Work

2.1 Geometric Constraints in Parameterization

Mesh parameterization has a vast and diverse literature, here we only refer to the comprehensive survey [Hormann et al. 2007] and the references therein. However, we know about only a few, isolated works that consider geometric constraints. [Bennis et al. 1991] and [Azariadis and Aspragathos 2001] investigate the problem of preserving the shape (geodesic curvature) of feature curves. [Wang 2008] and [Igarashi et al. 2009] give methods that preserve the length of curves in the context of cloth and garment design. [Vallet and Lévy 2009] extend the ABF algorithm to handle geometric constraints such as (C1). [Yin et al. 2008] map boundary curves to circles with a conformal parameterization – however, this only applies to boundary curves and a fixed domain (i.e. fixed circles). [Myles and Zorin 2012] developed a global parameterization method also capable of constraining isolines [Myles and Zorin 2013], this may

*e-mail:s.high.hopes@gmail.com
†e-mail:varady@iit.bme.hu

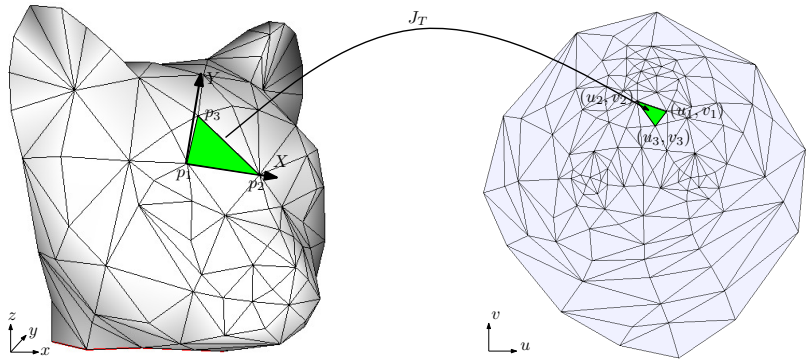


Figure 1: Mesh parameterization: affine functions (J_T) map each triangle from a reference position (in local coordinate system $X-Y$) to its final position in the $u-v$ plane.

be modified to handle more general constraints, such as (C1) and (C3) – handling constraints such as (C2) or (C4), however, requires constraining the relative scale of edges, which problem has not been addressed by said work. [Lipman 2012] and [Kovalsky et al. 2014] control the singular values of a parameterization using convex programming – this can be used to enforce constraints like (C5).

2.2 Constrained deformation and fitting

The problem of enforcing geometric constraints has received much attention in the actively developing field of constrained mesh deformation, see the recent survey [Mitra et al. 2013] and the references therein. A recent work, close to ours in spirit, is that of [Bouaziz et al. 2012], where certain kinds of geometric constraints are enforced during the deformation process: first, they fit each constrained subset independently with the required shape, then merge these mesh elements together in a global optimization step. The constrained fitting problem [Benko et al. 2002], arising in the reverse engineering of CAD-models, is another related research field.

2.3 Evaluation of Known Parameterization Methods

The enforcement of geometric constraints requires a parameterization method which allows us to prescribe both the relative orientation and the scale of mesh edges. In terms of vertex positions such constraints lead to nonlinear, even transcendent equations – which might be prohibitively expensive to solve for practical problem sizes.

Angle-based methods, such as ABF [Sheffer and de Sturler 2001] might be capable of enforcing geometric constraints, as was already noted by [Vallet and Lévy 2009], but constraining the relative lengths of edges, as well as their angles, leads yet again to quadratic equations in terms of vertex coordinates (when the planar embedding is constructed from the optimized angles).

More recent conformal parameterization methods are based on a proper discrete notion of conformality and optimize for *edge scaling factors* [Ben-Chen et al. 2008; Springborn et al. 2008; Myles and Zorin 2012], which might make them appear promising from the viewpoint of enforcing geometric constraints. However, a conformal map of a surface to the plane is uniquely determined by the boundary conditions. Furthermore, when one computes a conformal map in such a strict sense, one can prescribe either the curvature of the boundary curve (i.e. the angle between adjacent boundary edges) or the length of the boundary edges – but not both. Thus, although we could declare interior edges as parts of the boundary and transform constraints into boundary conditions, constraints such as (C2), (C4) and even (C3) cannot be generally satisfied by a proper conformal parameterization. This limitation has already been observed by [Myles and Zorin 2013], who remark that the most natural way to extend such conformal parameterization methods is to exploit the connection between conformal maps and the smoothest direction field on the surface – which leads to a method somewhat similar to our approach.

3 Analysis of Geometric Constraints

Our aim is to reduce the semantic high-level constraints (C1)-(C5) mentioned in section 1, to low-level constraints involving typical optimization variables such as positions or angles. Constraints such as (C1), and (C3) can be decomposed immediately into a set of requirements involving angles between certain edges of the mesh. For (C2) we require that a closed sequence of edges shall map to a closed polyline with a circumscribed circle, i.e. a *cyclic polygon*. Assume that we have a closed loop made out of N edges on the

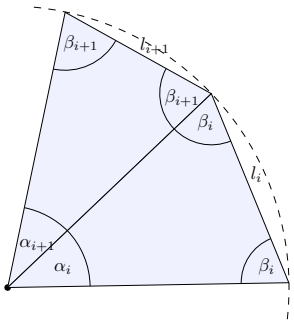


Figure 2: Notations for circle constraints.

mesh, with edge lengths $l_0, l_1, \ldots, l_{N-1}$. Now imagine that in the parameterization domain, said vertices lie on a circle, and each of the edges keeps its length or gets scaled by the same factor. For a *cyclic polygon* the side lengths divide the length of the whole polygon approximately in the same way as the respective sector angles divide 2π. Also observe that the triangle corresponding to each sector is equilateral. Then, given two neighboring edges l_i and l_{i+1} (see Figure 2):

$$\frac{l_i + l_{i+1}}{\sum_{i=0}^{N-1} l_i} = \frac{\alpha_i + \alpha_{i+1}}{2\pi}$$

$$\beta_i + \beta_{i+1} = \pi - \frac{\alpha_i + \alpha_{i+1}}{2}.$$

Substituting the former equation into the latter, we get the following for the interior angle between the edges:

$$\beta_i + \beta_{i+1} = \pi \left(1 - \frac{l_i + l_{i+1}}{\sum_{i=0}^{N-1} l_i} \right).$$

If the loop is actually a hole loop (interior boundary) of a multiply connected surface, we take the conjugate of these angles. We have made no assumption about the coplanarity of the edges, as for any set of edge lengths (that is possible on meshes), there exists a unique cyclic polygon [Pinelis 2005].

For (C4), i.e. for planar feature curves, the situation is trivial if the edges are exactly coplanar. In other cases, we fit a plane to the vertices, then project the vertices to the plane spanned by them for the angle calculations.

In summary, all of the constraints we have been considering can be reduced to some combination of the simple constraint that two edges shall span a prescribed angle and - excluding (C1) and (C3) - the ratio of their lengths shall be unchanged.

4 Parameterization with Geometric Constraints

We propose to compute a parameterization in two steps:

- First, we map the triangles individually to the plane, using rotations.
- Then, we 'stitch' the rotated triangles together into a planar mesh.

We will see, that our scheme has the advantage of allowing us to control both the relative orientation and the relative scale of mesh edges – this is how we improve upon previous similar methods proposed in the context of global parameterization and quadrilateral remeshing, which only considered constraining the isoline-structure of the parameterization and ignored the actual planar embedding. We elaborate on these steps in the following, along with details about the enforcement of geometric constraints.

4.1 Step 1 – Computing the Rotations

For the motivation of our initialization scheme, consider the naive way to compute a set of rotation matrices by isometrically flattening the 3D mesh on a per triangle basis, traversing a spanning tree of the faces starting from an arbitrary root. Obviously, such an approach would result in a highly non-optimal, even degenerate mapping after the subsequent 'stitching' phase, as the Gaussian curvature at a vertex is equal to the angular defect for the corresponding triangle fan, i.e. the amount by which the fan fails to close up after isometric flattening. This can be remedied by applying an appropriate amount of extra rotation on each triangle during the traversal of the spanning tree, with the aim of distributing the Gaussian curvature evenly in each triangle fan, see Figure 3. The criterion that in each inner triangle fan (and for multiply connected surfaces, around each homology generator) the net effect of the rotations shall counteract the Gaussian curvature can be expressed as an underdetermined set of linear equations for the rotation angles, of which we want to compute the solution with the smallest ℓ^2-norm.

$$\begin{aligned} \underset{\omega_{ij},\, ij \in E}{\text{minimize}} \quad & \|\omega\|_2^2 \\ \text{subject to} \quad & d_0^{\mathrm{T}} \omega = K, \end{aligned}$$

where d_0 is the vertex-edge adjacency matrix, ω is the vector of unknown rotation values for each (dual) edge, and K is the vector of Gaussian curvatures (for the interior vertices and homology generators). We note that this is practically equivalent to the so-called 'trivial connection' method for finding the smoothest direction field on a triangle mesh [Crane et al. 2010].

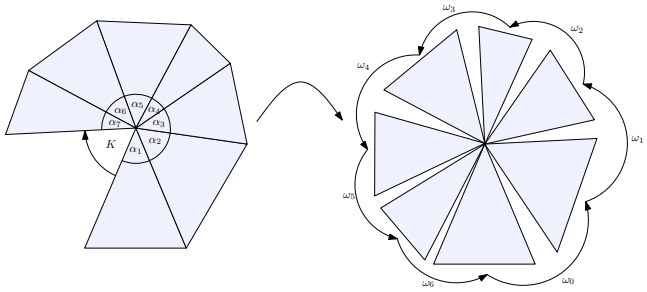


Figure 3: Interpretation of Step 1.

Enforcing Geometric Constraints

As we have an underdetermined linear system we can add any linear equations or split one of the existing ones.

- If two adjacent edges are required to make a prescribed angle, for interior vertices we split the corresponding equation, requiring that the rotations for the dual edges between the two constrained ones result in the given alignment, while assigning constant zero value to the rotations over the constrained edges, see Figure 4. For boundary vertices we simply add an analogous additional constraint to the system. We have observed that for multiply connected meshes, constraints involving inner boundary curves might conflict with the ones prescribing zero curvature. Thus, we omit the default constraint for inner boundary loops in the presence of a user-defined one.

- When two separate edges are required to span a prescribed angle, we build a dual path (by simple breadth-first search) between them and constrain the sum of rotations accordingly. More precisely, refer to Figure 5: assume that we have two (oriented) edges e_1 and e_2 constrained to span an angle φ, belonging to faces f_1 and f_2, which have, as their basis vectors the (oriented) edges b_1 and b_2. If α and β are the angles between e_1 and b_1 and e_2 and b_2 respectively, our constraint can be expressed as $\beta - \alpha' = \varphi$, where α' is the angle between b_2 and e_1. Along the chosen dual path b_2 is rotated with respect to b_1 by the angles dictated by the isometric flattening, denoted by ω and by the additional unknown rotations we apply along the traversed dual edges. Thus, as vectors transform with the inverse of coordinate transforms, these rotations have to be *subtracted* from α. So, in summary: $\alpha' = \alpha - \sum \omega$, and after separating the constants and the unknowns, our constraint becomes the following:

$$\sum_{\text{(dual path)}} \omega_{add.} = \varphi - \beta + \alpha - \sum_{\text{(dual path)}} \omega_{isom.}.$$

- For the preservation of planar regions, when an edge belongs to two constrained faces, we consider the corresponding rotation as constant zero and simply remove them from the optimization problem. It might appear natural to do the same for the edges on the boundary of the constrained region, but in our experience doing so causes numerical stability problems when solving the equations.

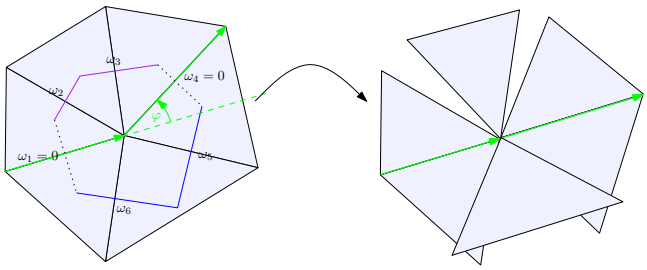


Figure 4: Notations for constraints involving two adjacent edges. The effect is shown on the right, when the prescribed angle is $\varphi = 0$.

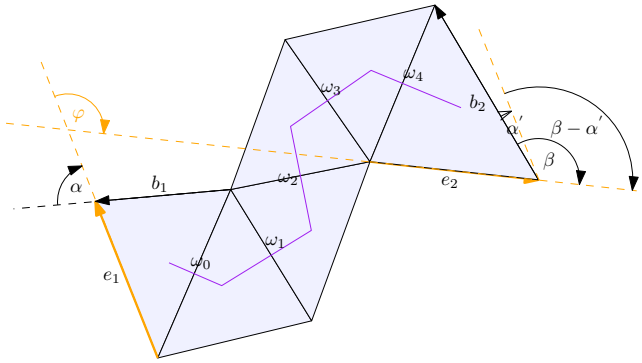


Figure 5: Notations for constraints involving two separate edges.

It could easily happen that the prescribed set of constraints lead to an infeasible problem and thus a contradictory set of equations, e.g. when two edges shared by a triangles are required to make a certain angle. Such problems however are readily alleviated, by simply refining the mesh in the vicinity of the problematic regions.

4.2 Step 2 – Computing a Planar Mesh from the Rotations

Next, we want to find a mapping of the mesh to the plane, which has as its Jacobians, matrices that are as close to the rotations (in Frobenius norm) as possible. This means solving the following *least-squares problem* for the vertex positions u and v:

$$\underset{u,v}{\text{minimize}} \quad \sum_T \|J_T(u,v) - R_T\|_F^2 \quad . \tag{1}$$

This is essentially the same energy that is used in As-Rigid-As-Possible parameterization [Liu et al. 2008] and a variety of vector field-based global parameterization and quadrilateral remeshing methods [Kälberer et al. 2007; Bommes et al. 2009; Myles and Zorin 2012]. It is known that the minimization of this quadratic form is equivalent to solving a pair of *Poisson equations*:

$$Lu = b_u$$
$$Lv = b_v,$$

where L is the well-known cotangent discretization of the Laplacian [Pinkall and Polthier 1993], while b_u, b_v are the (discrete) divergences of the rows of the rotation matrices - see e.g. Liu et al. [Liu et al. 2008] for explicit formulas.

Enforcing Geometric Constraints

Geometric constraints are actually enforced in the previous phase (by computing proper rotations); in the second phase, the only thing we can do (without needing expensive non-linear optimization), is to ensure that constrained edges get mapped to the plane – not just in a least-squares sense – but with *exactly* the given rotation matrices. Actually what one would actually want is that all edges involved in a given constraint should get mapped with *the same multiple* of their corresponding rotation matrices, to ensure that the constrained shape keeps its shape (for line constraints (C1) each constrained edge can have different scaling factors). It is one of our important observations that we can also include these multipliers (denoted by α) into the optimization, resulting in a set of equations such as the following:

$$u_j - u_i = \alpha_{ij}\left(R_{11}\left(x_j - x_i\right) + R_{12}\left(y_j - y_i\right)\right)$$
$$v_j - v_i = \alpha_{ij}\left(R_{21}\left(x_j - x_i\right) + R_{22}\left(y_j - y_i\right)\right),$$

which gives us an underdetermined linear system:

$$C \begin{bmatrix} u \\ v \\ \alpha \end{bmatrix} = d.$$

If we have preferred scaling factors, we can take this into account by adding weighted regularization-type terms to our quadratic objective:

$$\sum_{(ij)} w_{ij}\left(\alpha_{ij} - a_{ij}\right)^2,$$

where a_{ij} is the prescribed scaling factor, w_{ij} is the weight of the energy term associated with the edge e_{ij}. In what follows, we denote the diagonal matrix containing the weights by W and the set of preferred scaling factors by a.

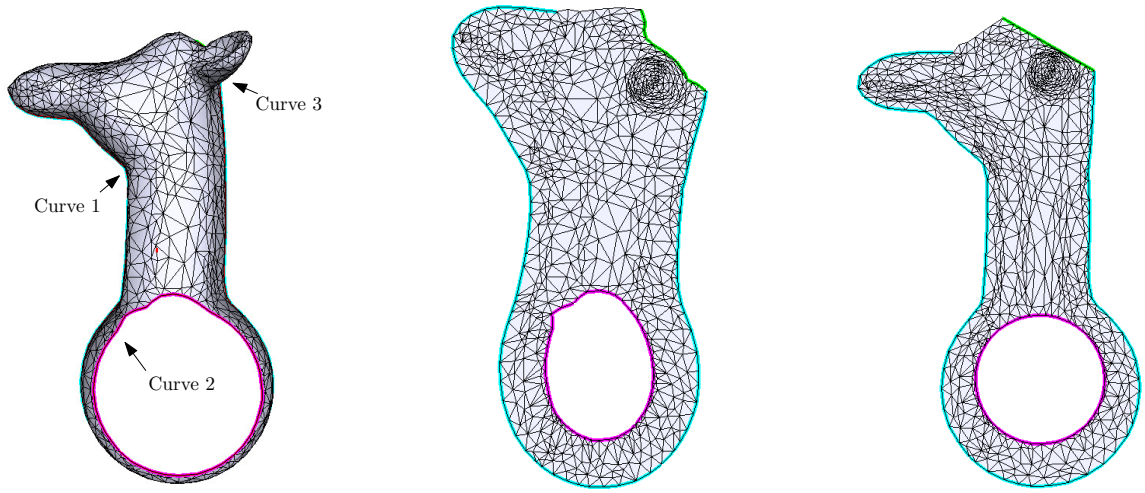
The resulting equality-constrained quadratic program can be solved via the method of Lagrange multipliers, i.e. by solving the following (indefinite) linear system:

$$\left[\begin{array}{cc|c} L & & C^{\mathrm{T}} \\ & L & \\ & & W \\ \hline C & & 0 \end{array}\right] \left[\begin{array}{c} u \\ v \\ \alpha \\ \hline \lambda \end{array}\right] = \left[\begin{array}{c} b_u \\ b_v \\ a \\ \hline d \end{array}\right].$$

Note that computing the rotations in Step 1 and enforcing them via the linear system above is a procedure that is completely independent of the energy we use in Step 2 – instead of (1), one could choose any other (quadratic) functional of the vertex positions.

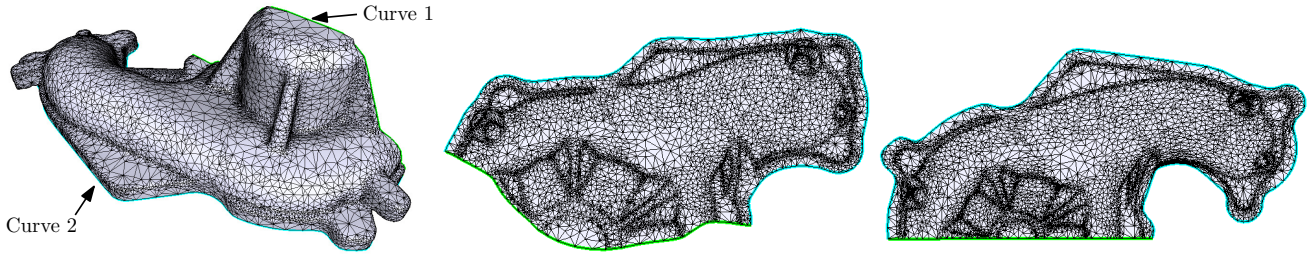
Enforcing Local Injectivity

Unfortunately, simply solving the linear system above does not necessarily yield a valid result in many cases. First, the scaling factors are not guaranteed to be positive. Second, as a cotangent Laplacian is used to fit a valid parameterization to the rotations, the resulting map might not be (locally) injective, i.e. certain triangles might reverse their orientations. There are ways to directly ensure that triangles do not flip during the parameterization, either by exploiting the injectivity of convex-boundary barycentric mappings [Kós and Várady 2003] or by resorting to more sophisticated nonlinear optimization [Lipman 2012; Bommes et al. 2013; Schüller et al. 2013]. We propose a much simpler *weighted least-squares* heuristic that is also capable of resolving these problems in most practical cases. The basic idea is to assign a weight to each triangle in the fitting energy, and should a triangle reverse its orientation in the parameterization, we increase its associated weight and repeat the fitting step with the new Laplacian matrix. We have found that increasing the weights in increments of 5 allow us to compute a (locally) injective constrained parameterization for the cost of no more than $20-40$ global iterations. We apply an identical scheme to the weights of scaling factors – in that case, the weights are multiplied by 10 in each iteration.



Model with constraints | Param. without constraints | Param. with constraints

Figure 6: Results for Test Model 1.



Model with constraints | Parameterization without constraints | Parameterization with constraints

Figure 7: Results for Test Model 2 (after 3 triangle weight iterations).

5 Results

5.1 Implementation

Our C++ implementation is built upon the OpenMesh library [Botsch et al. 2002]. The underdetermined and KKT systems are solved respectively with SuiteSparseQR [Davis 2011] and CHOLMOD [Chen et al. 2008].

5.2 Test Examples

We illustrate the capabilities of our method through several examples. These represent decimated, low resolution meshes, thus the reader can observe the mapping of triangles from 3D to 2D. In real applications, of course, meshes with much higher densities are used. The constraints were chosen on the basis of being illustrative, not in the context of any specific application. We stress, that we only prescribe the *geometric shape* of the constrained regions, their (relative) *scale* is optimized by our algorithm (for lines, each edge is scaled independently). Note: the prescribed scaling factors and weights were all set to 1, unless otherwise noted.

Figure 6, shows results for a multiply connected model of low resolution – Curve 1 is to be preserved, while the inner boundary Curve 2 is mapped to a circle, and Curve 3 to a line. For Figure 7, we require that Curve 1 is mapped to a line, while Curve 2 is planar and preserves its shape, thus effectively retaining a highly non-convex shape for the boundary. For Figure 8, the planar Region 1, and boundary Curve 1 shall preserve their shapes, while Curves 2 to 11 shall map to lines, forming an appropriate set of rectangles aligned with each other and to the boundary. For Figure 9, the planar Region 1 and the planar Curve 1 are to be preserved, while Curve 2 and 3 are to be mapped to circles.

5.3 Discussion

Our method provides a good framework for constrained parameterization; in the second phase, we can enforce any constraint that can be expressed as linear equations involving vertex positions. Adding constraints formulated by linear inequalities might lead to a quadratic program which could be solved efficiently with modern interior-point methods. Also, as we have remarked earlier, our method is completely generic with regards to the minimized energy or distortion measure – only the energy minimized in Step 2 needs to be changed to incorporate geometric constraints into other parameterization methods (assuming they directly optimize the vertex positions).

Triangle flips are currently prevented by our simple weighted least-squares heuristic; more general methods for computing (locally) injective parameterizations (a non-convex problem) either apply certain approximations to arrive at a convex optimization problem [Lipman 2012; Bommes et al. 2013] or require a flip-free mapping as a starting point [Schüller et al. 2013]. An alternative heuristic solution has been proposed recently by [Martinez Esturo et al. 2014], based on smoothing the minimized energy. Finding the method giving an acceptable compromise between robustness and efficiency is subject of future work.

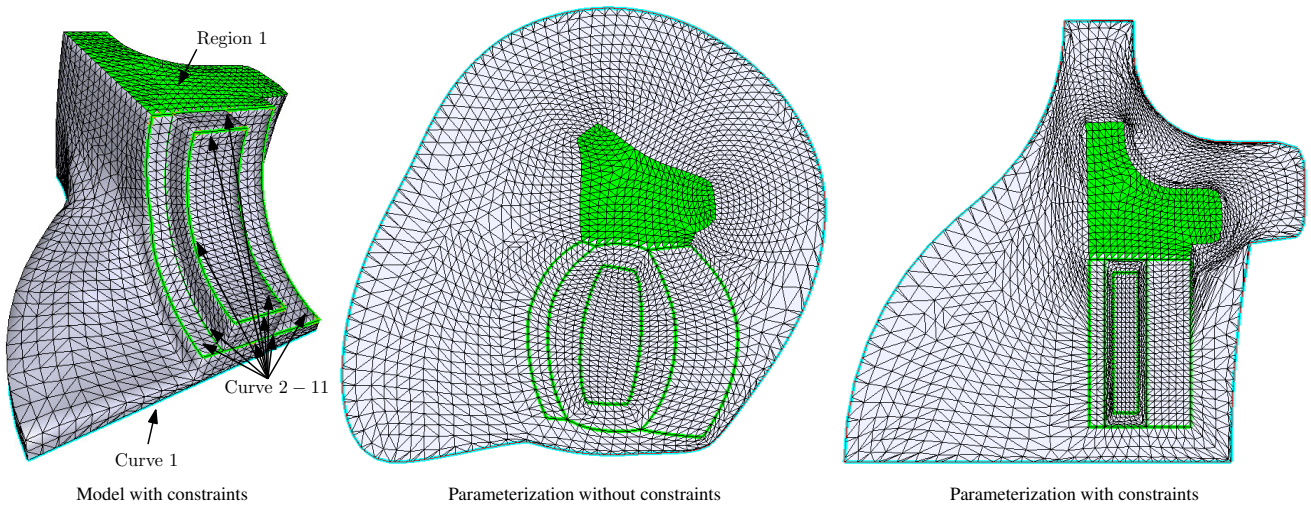


Figure 8: Results for Test Model 3 (Fandisk) (after 21 triangle weight iterations).

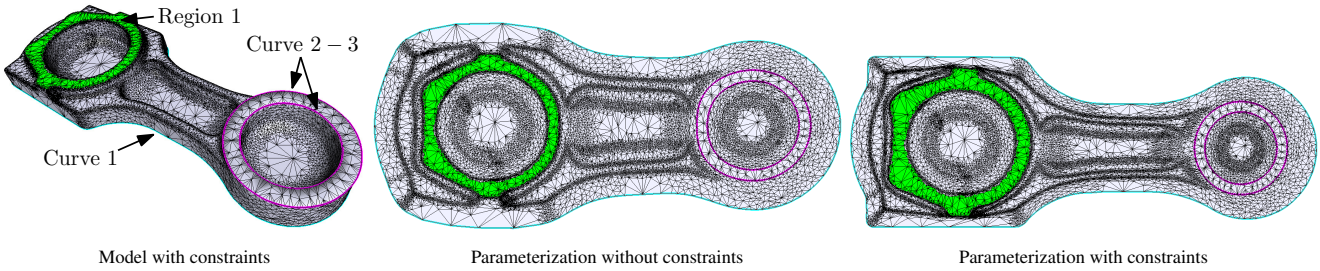


Figure 9: Results for Test Model 4 (after 3 triangle weight iterations).

6 Conclusion and Future Work

We have presented a framework capable of enforcing geometric constraints in the course of parameterizing triangle meshes and provided a more general solution than previous approaches. By means of this method various geometric features, such as, lines, circular arcs and various subregions can be constrained, while the ARAP energy is also minimized. The method can easily be adapted to parameterization methods that minimize alternative distortion measures.

It opens up interesting avenues for future research, where our approach is combined with methods recently proposed in the context of constrained deformation [Tang et al. 2014] or shape-space exploration [Yang et al. 2011]. We would also like to apply our algorithms to several interesting problems, such as controlled approximation of meshes using trimmed splines or transfinite surfaces and investigate constraint systems requested by different engineering applications.

References

AZARIADIS, P. N., AND ASPRAGATHOS, N. A. 2001. Geodesic curvature preservation in surface flattening through constrained global optimization. *Computer-Aided Design 33*, 8, 581–591.

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

BENKO, P., KÓS, G., VÁRADY, T., ANDOR, L., AND MARTIN, R. 2002. Constrained fitting in reverse engineering. *Computer Aided Geometric Design 19*, 3, 173–205.

BENNIS, C., VÉZIEN, J.-M., AND IGLÉSIAS, G. 1991. Piecewise surface flattening for non-distorted texture mapping. In *ACM SIGGRAPH computer graphics*, vol. 25, ACM, 237–246.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 77.

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

BOTSCH, M., STEINBERG, S., BISCHOFF, S., AND KOBBELT, L. 2002. OpenMesh: A Generic and Efficient Polygon Mesh Data Structure. In *OpenSG Symposium 2002*.

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.

CHEN, Y., DAVIS, T. A., HAGER, W. W., AND RAJAMANICKAM, S. 2008. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS) 35*, 3, 22.

CRANE, K., DESBRUN, M., AND SCHRÖDER, P. 2010. Trivial connections on discrete surfaces. In *Computer Graphics Forum*, vol. 29, Wiley Online Library, 1525–1533.

DAVIS, T. A. 2011. Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Transactions on Mathematical Software (TOMS) 38*, 1, 8.

ECKSTEIN, I., SURAZHSKY, V., AND GOTSMAN, C. 2001. Texture mapping with hard constraints. In *Computer Graphics Forum*, vol. 20, Wiley Online Library, 95–104.

HORMANN, K., LÉVY, B., AND SHEFFER, A. 2007. Mesh parameterization: Theory and practice. SIGGRAPH Course Notes.

IGARASHI, Y., IGARASHI, T., AND SUZUKI, H. 2009. Interactive cover design considering physical constraints. *Computer Graphics Forum 28*, 7, 1965–1973.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. QuadCover-Surface Parameterization using Branched Coverings. In *Computer Graphics Forum*, vol. 26, Wiley Online Library, 375–384.

KÓS, G., AND VÁRADY, T. 2003. Parameterizing complex triangular meshes. In *Curve and surface design*, Nashboro Press, T. Lyche, M.-L. Mazure, and L. L. Schumaker, Eds., 265–274.

KOVALSKY, S., AIGERMAN, N., BASRI, R., AND LIPMAN, Y. 2014. Controlling Singular Values with Semidefinite Programming. In *ACM Transactions on Graphics (TOG)*, ACM.

KRAEVOY, V., SHEFFER, A., AND GOTSMAN, C. 2003. *Matchmaker: constructing constrained texture maps*, vol. 22. ACM.

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

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. J. 2008. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, vol. 27, Wiley Online Library, 1495–1504.

MARTINEZ ESTURO, J., RÖSSL, C., AND THEISEL, H. 2014. Smoothed Quadratic Energies on Meshes. *ACM Transactions on Graphics (TOG) 34*, 1, 2.

MITRA, N., WAND, M., ZHANG, H. R., COHEN-OR, D., KIM, V., AND HUANG, Q.-X. 2013. Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, ACM, 1.

MYLES, A., AND ZORIN, D. 2012. Global parametrization by incremental flattening. *ACM Transactions on Graphics (TOG) 31*, 4, 109.

MYLES, A., AND ZORIN, D. 2013. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics (TOG) 32*, 4, 105.

PINELIS, I. 2005. Cyclic polygons with given edge lengths: existence and uniqueness. *Journal of Geometry 82*, 1-2, 156–171.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics 2*, 1, 15–36.

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

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

SPRINGBORN, B., SCHRÖDER, P., AND PINKALL, U. 2008. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics (TOG) 27*, 3, 77.

TANG, C., SUN, X., GOMES, A., WALLNER, J., AND POTTMANN, H. 2014. Form-finding with polyhedral meshes made simple. *ACM Transactions on Graphics (TOG) 33*, 4, 70.

VALLET, B., AND LÉVY, B. 2009. What you seam is what you get. Tech. rep., INRIA - ALICE Project Team.

WANG, C. 2008. WireWarping: A fast surface flattening approach with length-preserved feature curves. *Computer-Aided Design 40*, 3, 381–395.

YANG, Y.-L., YANG, Y.-J., POTTMANN, H., AND MITRA, N. J. 2011. Shape space exploration of constrained meshes. *ACM Trans. Graph. 30*, 6, 124.

YIN, X., DAI, J., YAU, S.-T., AND GU, X. 2008. Slit map: Conformal parameterization for multiply connected surfaces. In *Advances in Geometric Modeling and Processing*. Springer, 410–422.