Seventh Hungarian Conference on Computer Graphics and Geometry, Budapest, 2014

Mesh Parameterization with Geometric Constraints

Márton Vaitkus,[1] Tamás Várady[1]

[1] Budapest University of Technology and Economics

Abstract

Mesh parametrization is an important topic in computer graphics and 3D geometric modelling. Methods differ in how the distortion of the parameterized mesh is defined and what sort of optimization methods are applied to map the mesh into the domain in a computationally efficient manner. There exist several emerging applications where in addition to minimizing the distortion, we wish to prescribe the mapping of a few selected feature curves and feature regions, and their relationship. We will evaluate and compare existing methods focusing on their capability of enforcing various geometric constraints. Then we introduce two new algorithms that are suitable to satisfy these demands. The first is an extension of the As-Rigid-As-Possible approach, where we perform optimization and simultaneously maintain constraints. The second method is based on an iterative deformation of an existing mapping until the prescribed constraints get satisfied. Several examples illustrate our approaches comparing mappings without and with constraints.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling — triangular meshes, parameterization, constraints

1. Introduction

Parameterization, or 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 surfaces with disc topology, this mapping can be assumed 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 2. The distortion of the parameterization is quantified by considering these Jacobian matrices, which depend linearly on the vertex coordinates.[24]

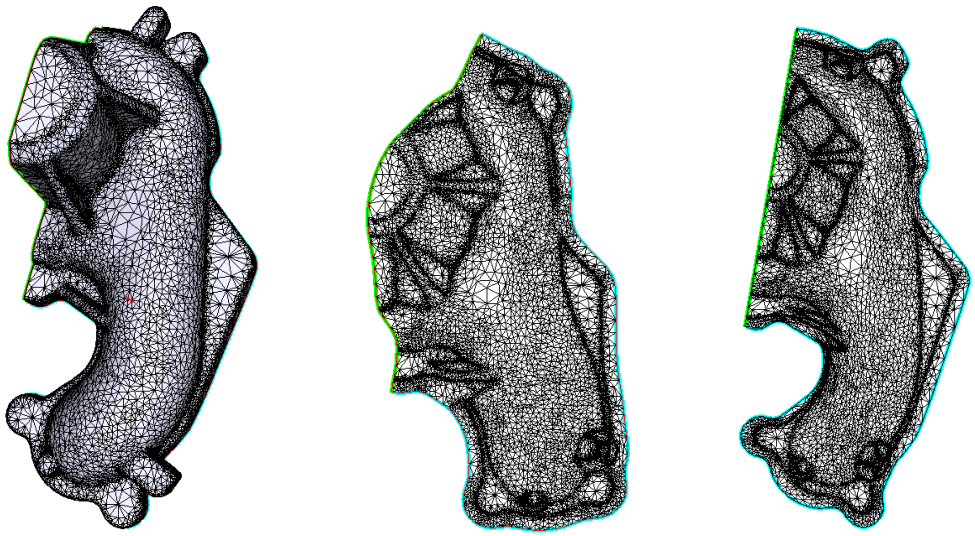
Up until now, the research community has been mainly concerned with minimizing the distortion of the parameterization. A huge variety of distortion measures have been proposed along with efficient algorithms for their minimization. Nevertheless, there exist practical applications, where the demand to 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[11,14] and constant coordinate lines,[5,20] 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 are 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 considered in such generality; and we are aware of only a few isolated attempts to enforce certain subsets of them.

Hereinafter we assume that the feature curves and the



(a) (b) (c)

Figure 1: Results for an automotive part - (a): Model with constraints (green curve is to be mapped to a line, cyan curve is a planar curve and must preserve its shape, (b): Unconstrained ARAP parameterization, (c): Constrained ARAP parameterization.

boundaries of the features regions are bounded by polylines lying on the mesh. These polylines may not necessarily run on the edges of the triangulation; in these cases a preprocessing phase is performed that splits the triangles involved and embed related edges into the mesh structure.

The paper is structured as follows. First we give an overview of the relevant literature (section 2), then analyze the aforementioned geometric constraints, with the aim of reducing them to elementary relations between mesh elements (section 3). Next, we evaluate known parameterization methods according to their potential, regarding the enforcement of geometric constraints (section 4). Our main contribution is presented in section 5, where we propose an extension to the As-Rigid-As-Possible approach, along with a novel initialization scheme. In section 6, we present another method to enforce geometric constraints on an already computed parameterization, by deforming the planar mesh iteratively.

2. Previous Work

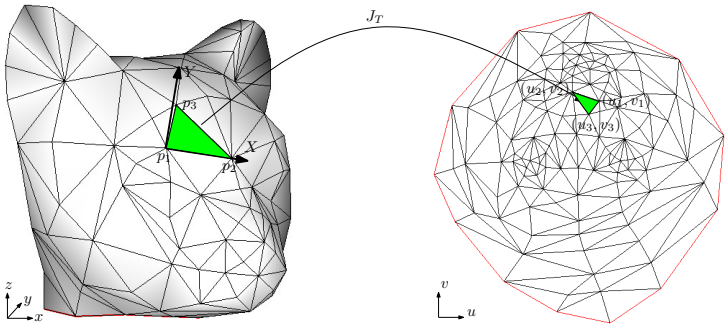
Mesh parameterization has a vast and diverse literature, here we only refer to the comprehensive surveys[24,25] and the references therein.

The majority of commonly used parameterization methods aims at angle preservation, i.e. they compute an optimized *conformal* mapping. This can be done by minimizing (typically quadratic) energies involving vertex positions,[10,15] angles[23,30] or edge lengths.[2,9,19,27]

More general distortion measures typically necessitate computationally expensive non-linear optimization. A notable exception is the application of the As-Rigid-As-Possible mesh deformation algorithm[26] to mesh parameterization,[17] which computes an optimized isometric mapping using an efficient iterative procedure.

We know about only a few, isolated works in the parameterization literature that consider some kinds of geometric constraints. Bennis et al.[4] and Azariadis et al.[1] investigate the problem of preserving the shape (geodesic curvature) of feature curves. Wang et al.[29] and Igarashi et al.[12] give methods that preserve the length of curves in the context of cloth and garment design. Chen et al.[16] preserve of shape of feature curves in a finite element simulation of elastic flattening. Valet and Levy[28] extends the ABF algorithm to handle geometric constraints such as (C1).

Although geometric constraints are relatively new to the subfield of mesh parameterization, they have been thoroughly investigated and applied in geometric modeling. The enforcement of geometric constraints, known as *constrained modeling*, is a quintessential part of the majority of computer-aided design systems. Constrained modeling has a very voluminous literature, and is a topic of ongoing research; we refer to the survey[13] and the references therein. An important difference between our approach and traditional constrained modeling techniques is that we work with unorganized triangle meshes, not the organized models made up of larger scale geometric primitives common in CAD-systems. The constrained fitting problem,[3] arising in the reverse engineering of CAD-models, is another related research field.



(a)

Figure 2: Illustration of mesh parameterization. The parameterization is defined by the vertex coordinates and is a collection of affine functions (J_T), mapping each triangle from a reference position (in local coordinate system $X-Y$) to its final position in the $u-v$ plane.

More closely related is the actively developing field of constrained mesh deformation, see the recent survey[18] and the references therein. Our approach is novel in the sense that some of the features we are considering are not required to be present in the original model in the desired form. One recent work, close to ours in spirit, is that of Bouaziz et al.,[6] where certain kinds of geometric constraints are enforced during the deformation process by a generalization of the local-global algorithm for the minimization of the ARAP energy:[26] first, they fit each constrained subset independently with the required shape, then merge these mesh elements together in a global optimization step.

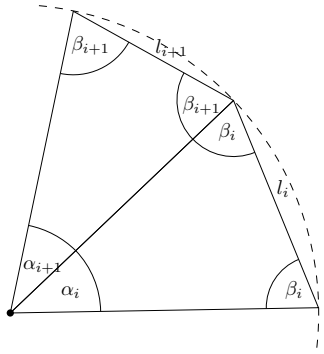


Figure 3: Notations for the circle constraints.

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. To do the same for (C2) and (C4) we will need some non-trivial arguments. 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 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 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 3):

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

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

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.[21]

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.

Constraint (C5) usually requires special care, in a way that, as we will see later, depends on the employed parameterization method.

In summary, all of the constraints we have been considering can be reduced to some combination of the following two kinds of low-level constraints:

- Two edges shall span a prescribed angle and - excluding (C1) and (C3) - their length ratio shall be unchanged.
- A triangle shall keep its shape, i.e. its angles and/or edge lengths.

4. Evaluation of known parameterization methods

Based on the results of the previous section, we can easily add geometric constraints to any parameterization method optimizing directly for vertex positions in the parameter plane, such as DNCP[10] or LSCM.[15] When we require that two edges (whether they share vertex in the mesh or not) shall span an angle in the parameterization, this is equivalent to demanding that one edge (e_{kl}) is the scaled and rotated version of the other (e_{ij}). The scaling can be arbitrary, but the most natural choice is the ratio of the edge lengths, which results in a curve similar to their original counterparts on the 3D mesh. Quantitatively this can be expressed as:

$$u_l - u_k = \frac{|e_{kl}|}{|e_{ij}|}(\cos(\varphi)(u_j - u_i) - \sin(\varphi)(v_j - v_i))$$

$$v_l - v_k = \frac{|e_{kl}|}{|e_{ij}|}(\sin(\varphi)(u_j - u_i) + \cos(\varphi)(v_j - v_i))$$

where u_i, v_i are the coordinates of the vertex i, and φ is prescribed angle between the edges.

The shape of triangles can be preserved exactly by requiring the Cauchy-Riemann equations to be satisfied, but, for conformal methods this has a negligible effect usually, as the mappings have negligible angular distortion in developable regions by default.

We omit results for these algorithms as our implementation for the DNCP method gave degenerate results in cases involving constraints on the boundary. We conjecture that this is a result of a conflict between the Neumann boundary conditions and the constraints, but we have found no straightforward way to remedy the problem.

Angle-based methods, such as ABF[23,30] might appear capable to enforce geometric constraints, as was already noted by Vallet and Lévy,[28] but they have a fundamental limitation, namely that with the exception of (C1) and (C3), geometric constraints cannot be expressed purely in terms of angles. For example: an N-gon has $2N-4$ degrees of freedom, of which we can control only N via the interior angles.

Most of the known position and angle-based methods have a common, serious drawback: they are typically conformal methods, and thus, they only strive to preserve the mesh up to its angles, i.e. up to (local) similarity. As these methods are agnostic to scale, they allow us no direct, tractable control (e.g. expressible via linear equations in the optimization variables) over the metric properties of the parameterization.

We also note that almost all of the published distortion measures can be expressed as functions of the Jacobian singular values, which might appear to give us control over the local scale of the mapping and thus the scale of the features, but as the singular values are in a complicated nonlinear relationship with the actual optimization variables, this is not tractable for practical problem sizes.

There exist a wide class of differential geometric methods optimizing for edge length scale factors,[2,9,19,27] which might appear promising from the viewpoint of geometric constraints. It has turned out that although these methods seem to employ a variational principle, what they optimize is not a distortion measure, but an integrability condition, and their result is fundamentally unique up to the boundary conditions. Thus adding constraints involving the mesh interior would violate the mathematical assumptions underlying these methods and render the problem infeasible.

In summary, we conclude that the only algorithm that is feasible of incorporating all of the constraints under consideration in a computationally tractable way is the As-Rigid-As-Possible parameterization method.

5. As-Rigid-As-Possible Parameterization with Geometric Constraints

5.1. Baseline algorithm

The algorithm based on the iterative minimization As-Rigid-As-Possible (ARAP) energy was first proposed by Liu et al.[17] It builds on earlier work on mesh deformation by Sorkine and Alexa.[26] The proposed distortion measure, the *ARAP energy* measures, per triangle, the distance (in Frobenius norm) of the parameterization (represented by a 2-by-2 Jacobian matrices in local coordinate frames) from a local isometry, i.e. a rotation matrix:

$$\begin{array}{ll} \underset{\text{for } J_T, T \in F}{\text{minimize}} & \sum_{T \in F} A_T \left\| J_T - R_T \right\|_{\mathrm{F}} \\ \text{subject to} & R_T \in SO(2) \end{array}$$

where J_T is the Jacobian matrix for the triangle T.

This is a highly nonlinear, non-convex problem, a locally optimal solution of which can be nonetheless computed quite efficiently using the so-called *local-global algorithm*. Notice that if we keep one term of the energy fixed, while optimizing for the other, the problem simplifies:

- Given a fixed set of Jacobians, i.e. a parameterization, one can compute the closest set of rotations on a per triangle

basis using a simple closed-form formula. As this can be done independently, even parallel for each triangle, this is called the *Local phase*.

- Given a set of rotation matrices, one can fit a set of Jacobians to it in a least squares sense by solving a pair of Poisson equations for the vertex positions:

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

where L is the usual cotangent Laplacian (i.e. a weighted vertex-vertex adjacency matrix, depending only on the original $3D$ geometry), and b_u (resp. b_v) is the discrete divergence for the vector field given by applying the rotation matrices to the u (resp. v) coordinates (see Liu et al. for details). This is called the *Global phase*.

By iterating between these two phases, one can find a local minima of the ARAP energy, for the effective cost of a single matrix factorization (as the coefficient matrix in the global phase stays constant through the iterations).

5.2. Initialization of rotation matrices

The iterative algorithm described above is usually initialized by computing a set of Jacobians, i.e. a parameterization using some other method, then proceeding with the first local phase. This is obviously difficult if geometric constraints are imposed, and furthermore — as it was observed by Myles and Zorin[19] — initializing the *rotation matrices* and proceeding with the first global step leads to significantly faster convergence in practice. We reformulate the basic idea of Myles and Zorin in a form more suitable to our goals, by appealing to the analogy with vector field design.

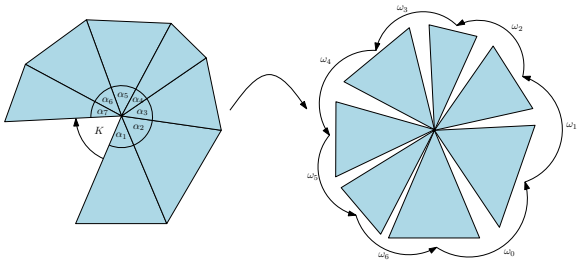


Figure 4: Interpretation of the ARAP initialization algorithm.

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 a arbitrary root. Obviously, such an approach would result in a highly non-optimal, even degenerate mapping after the subsequent global iteration, 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 4. The criterion that in each inner triangle fan 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[†], a standard optimization problem:

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

where d_0 is the vertex-edge adjacency matrix (the exterior derivative for 0-forms[7]), ω is the vector of unknown rotation values for each (dual) edge, and K is the vector of Gaussian curvatures (for the interior vertices).

5.3. Geometric constraints

Geometric constraints must be enforced in each of the iterations. Observe that the actual shape and orientation of the constrained features are determined during initialization, they are just preserved or scaled in the subsequent local and global phases (which is the main reason why one must initialize the rotation matrices instead of the Jacobians).

5.3.1. Initialization phase

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 5. For boundary vertices we simply add an additional constraint. 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 hole 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 6: as-

† We note that this is practically equivalent to the method of Crane et al.[8] for computing a *trivial connection*, represented as a discrete differential dual 1-form, i.e. a scalar function on the dual edges, that is the closest to the canonical *Levy-Civita connection* of the surface, for the purposes of vector field generation.

sume 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 we have run into numerical stability problems while doing so.

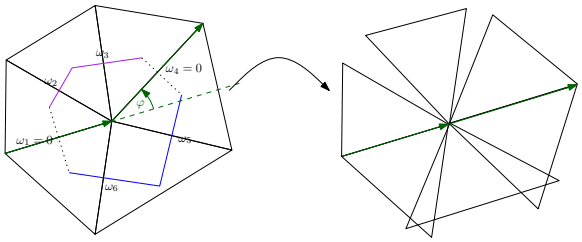


Figure 5: Notations for constraints involving two adjacent edges in ARAP. On the right we illustrate the effect of the constraint, if the prescribed angle is $\varphi = 0$.

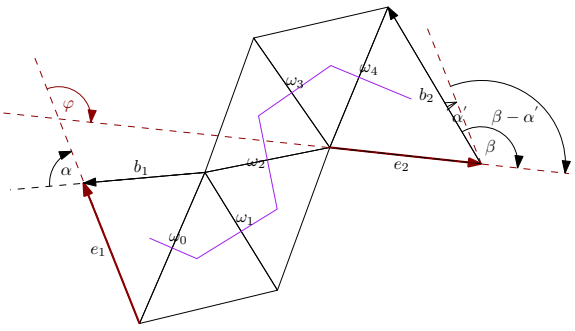


Figure 6: Notations for constraints involving two separate edges in ARAP.

5.3.2. Global phase

First, we recast the problem of solving the linear systems approximating the Poisson equations as minimizations of quadratic forms, then we enforce linear constraints by the use of *Lagrange multipliers* so the optimization problem for the u (resp. v) coordinates

$$\begin{array}{ll} \underset{\text{for } u \in \mathbb{R}^{N_V}}{\text{minimize}} & u^T L u - b_u^T u \\ \text{subject to} & Cu = d_u \end{array}$$

is solved via the symmetric, indefinite linear system

$$\begin{bmatrix} L & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ \nu \end{bmatrix} = \begin{bmatrix} b_u \\ d_u \end{bmatrix}.$$

For edge-based constraints, we simply require that the given edges are mapped to the plane exactly with the corresponding rotation matrices. The same constraints could, in principle be used on the edges of preserved regions, but we have found that it is numerically more stable to constrain the Jacobian of each triangle to be equal to the corresponding rotation.

Note that these constraints enforce isometry for said edges or triangles, but the user has the freedom to prescribe arbitrary scaling factors for each feature independently. This approach is not without its drawbacks, however, examine the task shown in Figure 1, where a planar curve constraint and straight line mapping is prescribed. It is obvious, that it is impossible to satisfy both constraints by mapping the boundary edges isometrically. Although one could find appropriate scaling factors heuristically, the general solution is to allow constraints that enforce geometric properties only up to similarity as done in position-based methods, which, however creates a coupling between the u and v coordinates, resulting in a much larger linear system.

As a cotangent Laplacian is used to fit a valid parameterization to the rotations, the resulting map might not be one-to-one. This could happen for a single triangle, or what is more common in the presence of geometric constraints, an entire region could 'spill over' a highly curved, concave boundary. Although there are ways to prevent such artifacts explicitly,[22] we have implemented a less expensive ad-hoc procedure instead: if a triangle reverses its orientation, we modify the Laplacian matrix by adding a positive weight to the entries that correspond to said triangle, and repeat the global phase with the updated matrix, iterating until there are no flipped triangles. With this method we have managed to avoid overlaps completely in all of our examples.

We have also experimented with the more refined weighting scheme in,[5] called *local stiffening*. We have found that while this procedure might be well-suited for the prevention of local, isolated overlaps, it fails to converge in a reasonable time for the more expansive 'spills' we have encountered.

5.3.3. Local phase

To preserve constraints enforced in a preceding phase, we simply ignore those triangles that are part of a preserved region or contain a bounded edge.

6. Enforcement of Geometric Constraints via Iterative Deformation

Previously, we have considered extensions to various parameterization methods. Our next aim is to enforce geometric constraints by the deformation of a previously created, already exisiting parameterization, i.e. a planar mesh. This can be achieved using a two-step iterative process: first isolate the constrained mesh elements and tweak them to the desired shape, then deform the rest of the mesh accordingly.

6.1. Deforming the constrained elements

If the shape is known beforehand, such as in the case of constraints (C4) and (C5), we simply fit said shape to the corresponding parts of the parameterization via standard procrustes analysis, i.e. computing the SVD of the covariance matrix, after removing the mean of the point-sets. In theory the same method could be applied to the cases (C1) and (C2), but instead we have implemented a more robust scheme based on the curvature flow of Crane et al.,[9] which is isometric by construction and is stable for extraordinarily large timesteps. This allows us to perform the deformation gradually in a natural way, enabling more refined user interactions.

6.2. Deforming the parameterization

Knowing the new positions of the constrained vertices, we can easily deform the entire 2D mesh accordingly, by applying the usual local-global ARAP iterations with the positions of the constrained vertices treated as constants in the global phase.

7. Results

Results with the ARAP algorithm for a variety of geometric constraints can be seen in Figure 1, Figure 7, Figure 8 and Figure 9. All results were obtained after initialization and a single global step, with the exception of those of Figure 1 where several global iterations were necessary to resolve overlaps at concave regions of the boundary.

8. Conclusions and Future Work

The enforcement of geometric constraints is an emerging new research area in the context of mesh parameterization. We have analyzed a wide class of geometric constraints and evaluated existing parameterization methods according to their capability to enforce them. We have found that a straightforward extension of the As-Rigid-As-Possible method is capable handling all of the constraints under consideration, and it can also be used to deform an existing parameterization to satisfy geometric constraints.

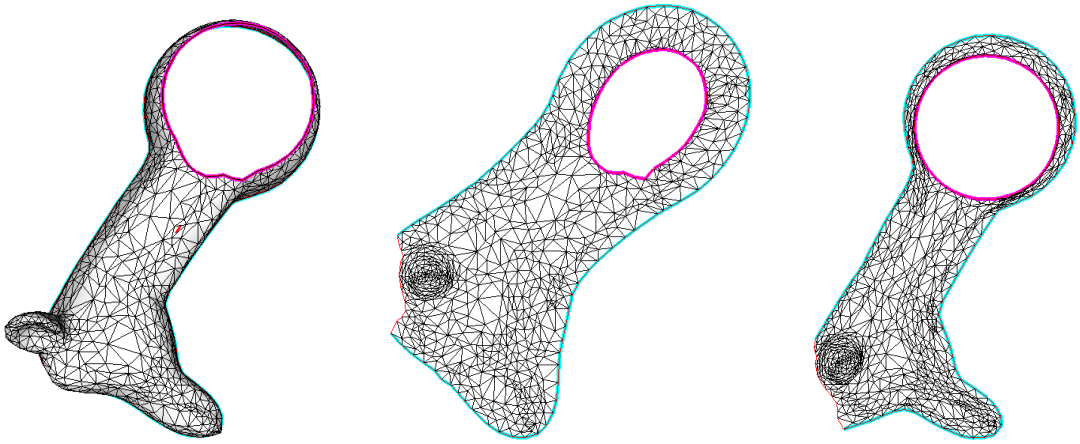
In the future we would like investigate further variations of position-based methods including algorithms based on eigenvalue computations, find ways to optimize edge scaling factors in an explicit way in the context of ARAP, and consider more general constraints e.g. ones represented by inequalities or nonlinear equations. We also plan to apply our algorithms to optimize the parameterization of data points for fitting free-form surfaces.

Acknowledgements

We are grateful to Péter Salvi and Kai Hormann for useful technical discussions. This project was supported by the Hungarian Scientific Research Fund (OTKA No. 101845).

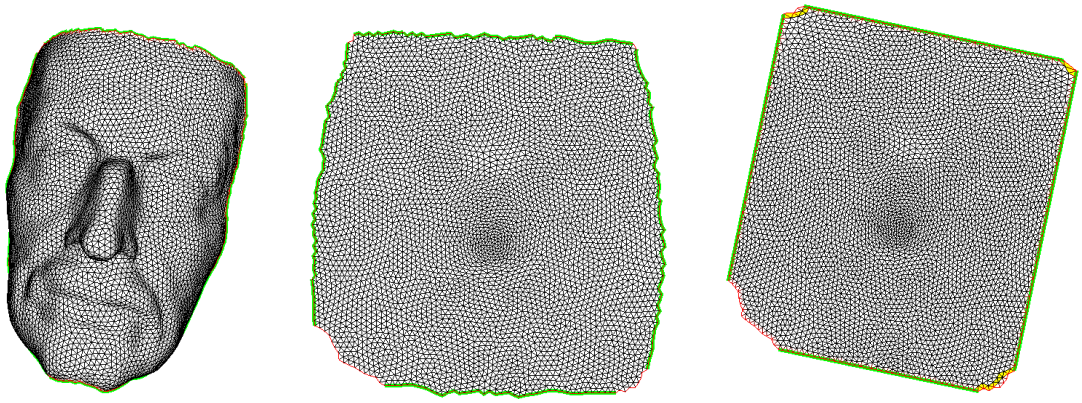
References

1. P. N. Azariadis and Nikos A. Aspragathos. Geodesic curvature preservation in surface flattening through constrained global optimization. *Computer-Aided Design*, 33(8):581–591, 2001.
2. M. Ben-Chen, C. Gotsman, and G. Bunin. Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum*, 27(2):449–458, 2008.
3. P. Benkö, G. Kós, T. Várady, L. Andor, and R. Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19(3):173–205, 2002.
4. C. Bennis, J. Vézien, and G. Iglésias. Piecewise surface flattening for non-distorted texture mapping. *ACM SIGGRAPH Computer Graphics*, 25(4):237–246, 1991.
5. D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics (TOG)*, 28(3):77, 2009.
6. S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly. Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum*, 31(5):1657–1667, 2012.
7. K. Crane, F. de Goes, M. Desbrun, and P. Schröder. Digital geometry processing with discrete exterior calculus. *ACM SIGGRAPH 2013 courses*, 2013.
8. K. Crane, M. Desbrun, and P. Schröder. Trivial connections on discrete surfaces. *Computer Graphics Forum*, 29(5):1525–1533, 2010.
9. K. Crane, U. Pinkall, and P. Schröder. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)*, 2013.



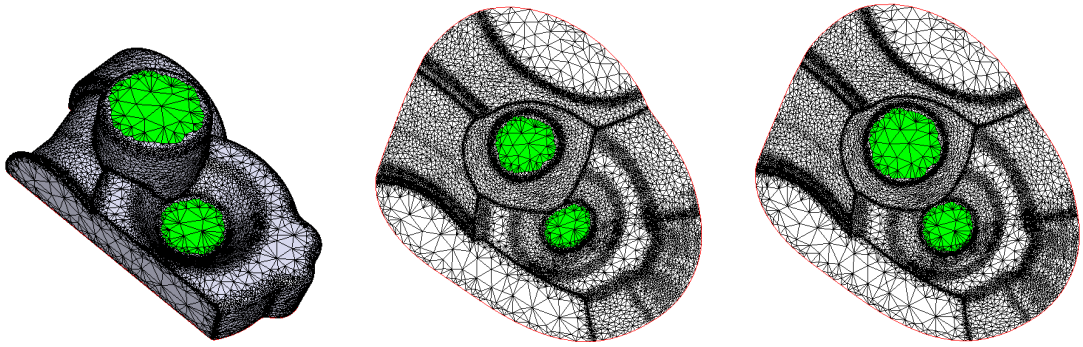
(a) (b) (c)

Figure 7: Results for Giraffe model - (a): Model with constraints (magenta curve is to be mapped to a circle, cyan curve is planar and must preserve its shape), (b): Unconstrained ARAP parameterization, (c): Constrained ARAP parameterization.



(a) (b) (c)

Figure 8: Results for Maxface model - (a): Model with constraints (green curves are to be mapped to a line, and be perpendicular), (b): Unconstrained ARAP parameterization, (c): Constrained ARAP parameterization.



(a) (b) (c)

Figure 9: Results for Darmstadt model - (a): Model with constraints (green planar regions are to preserve their shape), (b): Unconstrained ARAP parameterization, (c): Constrained ARAP parameterization.

10. M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2003.
11. I. Eckstein, V. Surazhsky, and C. Gotsman. Texture mapping with hard constraints. *Computer Graphics Forum*, 20(3):95–104, 2001.
12. Y. Igarashi, T. Igarashi, and H. Suzuki. Interactive cover design considering physical constraints. *Computer Graphics Forum*, 28(7):1965–1973, 2009.
13. C. Jermann, G. Trombettoni, B. Neveu, and P. Mathis. Decomposition of geometric constraint systems: a survey. *International Journal of Computational Geometry & Applications*, 16(05–06):379–414, 2006.
14. V. Kraevoy, A. Sheffer, and C. Gotsman. Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics (TOG)*, 22(3):326–333, 2003.
15. B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371, 2002.
16. Wen liang Ch., Peng W., and Yidong B. Surface flattening based on linear-elastic finite element method. 5(7):728 – 734, 2011.
17. L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. *Computer Graphics Forum*, 27(5):1495–1504, 2008.
18. N. Mitra, M. Wand, H. Zhang, D. Cohen-Or, and M. Bokeloh. Structure-aware shape processing. *Eurographics 2013-State of the Art Reports*, pages 175–197, 2012.
19. A. Myles and D. Zorin. Global parametrization by incremental flattening. *ACM Transactions on Graphics (TOG)*, 31(4):109, 2012.
20. A. Myles and D. Zorin. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics (TOG)*, 32(4):105, 2013.
21. I. Pinelis. Cyclic polygons with given edge lengths: existence and uniqueness. *Journal of Geometry*, 82(1-2):156–171, 2005.
22. C. Schüller, L. Kavan, D. Panozzo, and O. Sorkine-Hornung. Locally injective mappings. *Computer Graphics Forum*, 32(5):125–135, 2013.
23. A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers*, 17(3):326–337, 2001.
24. A. Sheffer, K. Hormann, and B. Lévy. Mesh parameterization: Theory and practice. *ACM SIGGRAPPH, course notes*, 2007.
25. A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.
26. O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *ACM International Conference Proceeding Series*, volume 257, pages 109–116. ACM, 2007.
27. B. Springborn, P. Schröder, and U. Pinkall. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics (TOG)*, 27(3):77, 2008.
28. B. Vallet and B. Lévy. What you seam is what you get. Technical report, INRIA - ALICE Project Team, 2009.
29. C. Wang. Wirewarping: A fast surface flattening approach with length-preserved feature curves. *Computer-Aided Design*, 40(3):381–395, 2008.
30. R. Zayer, B. Lévy, and H. P. Seidel. Linear angle based parameterization. In *Fifth Eurographics Symposium on Geometry Processing-SGP 2007*, pages 135–141, 2007.