Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Háromszöghálók paraméterezése geometriai kényszerek figyelembevételével

---

Triangle Mesh Parameterization with Geometric Constraints

Tudományos Diákköri Dolgozat/ Scientific Students' Association Paper

Készítette/Author
Vaitkus Márton

Konzulens/Supervisor
Dr. Várady Tamás

October 25, 2013

Contents

Introduction

Mesh parameterization is one of the fundamental operations in computer graphics and geometry processing. Initially motivated by the needs of texture mapping, it has since become indispensable for applications such as surface fitting, remeshing, mesh repair, deformation and simplification. The problem is inherently difficult, and for its analysis and solution one must utilize concepts from highly advanced mathematical fields such as differential geometry and complex analysis. As such, it has generated some of the most diverse and extensive research in the field, resulting in dozens of publications each year.

Parametric curves and surfaces are ubiquitous in computer graphics. In mesh parameterization, we would like to solve an inverse problem, as illustrated in Figure 1: given a surface, represented by a triangle mesh in $\mathbb{R}^3$, we would like to find a mapping into $\mathbb{R}^2$, i.e. the Euclidean plane. Although we would prefer to do so without any distortion, a fundamental result in differential geometry tells us, that this is not possible: surfaces cannot be mapped to the plane without distortion if their (Gaussian) curvature is not zero. This means that, in practice, some kind of compromise has to be made: distortion of angles, lengths or areas has be tolerated to a certain extent.

Historically, the original motivation for mesh parameterization has been texture mapping, where we would like to add some detail (color or geometry) to the mesh, using information stored in some kind of image.

Another problem, where a parameterization of the mesh could be necessary, is reverse engineering [Varady et al., 1997], or more precisely the *fitting* of triangle meshes with parametric surfaces. The challenging task of computing optimal control point and knot positions can be reduced to a regression problem - assuming that a suitable parameterization is available.

The generation of a quadrilateral mesh from a triangular one is a difficult task [Bommes et al., 2012]. Most of the state-of-the-art methods for quad remeshing are based on parameterization, as the coordinate lines of the plane can be sampled to define a quadrilateral cell structure.

Minimizing geometric distortion has been the focus of the majority of parameterization research, but for most practical applications it is necessary to adhere to additional constraints. For example, in the case of automated texture mapping, it is a natural requirement, that certain points are mapped to given locations in the parameter

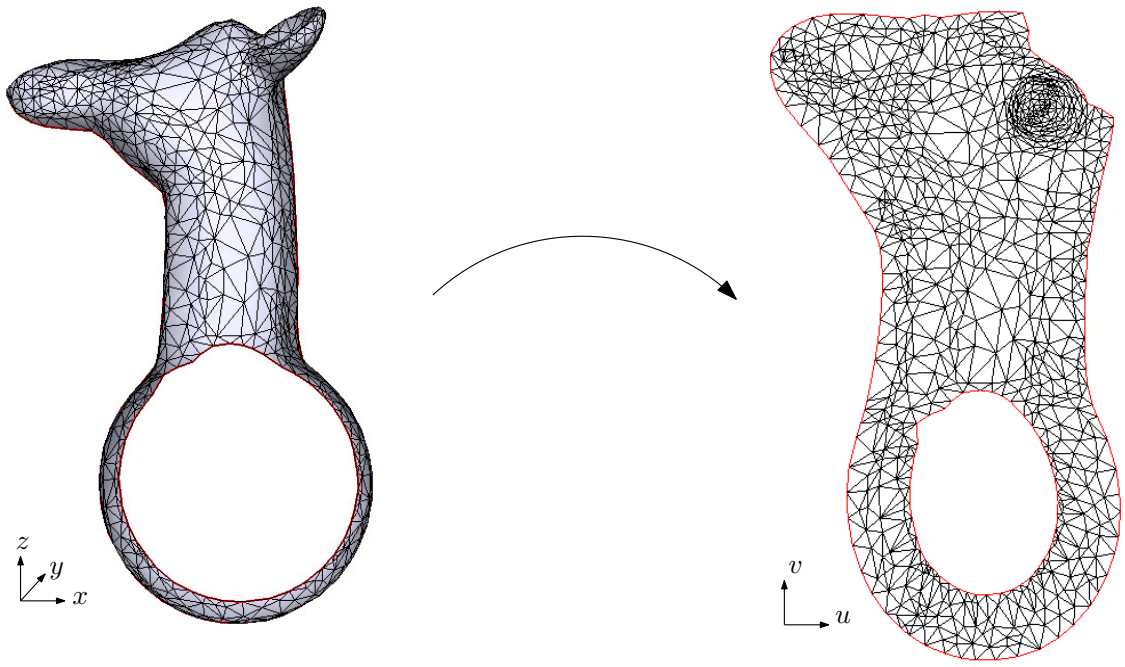


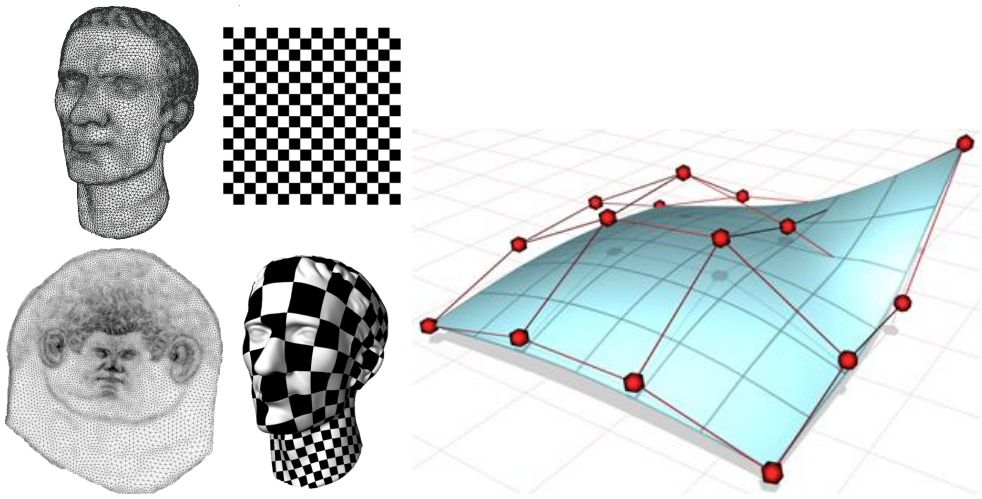
Figure 1: *Illustration of parameterization.*

domain [Eckstein et al., 2001], [Kraevoy et al., 2003]. Another popular research topic in constrained parameterization is motivated by quad mesh generation, where it is crucial that sharp edges and other salient mesh features are mapped to constant coordinate lines [Bommes et al., 2009], [Myles and Zorin, 2013]. The main topic of this paper concerns the enforcement of higher level, *geometric constraints* in parameterization methods. Examples of such constraints are:

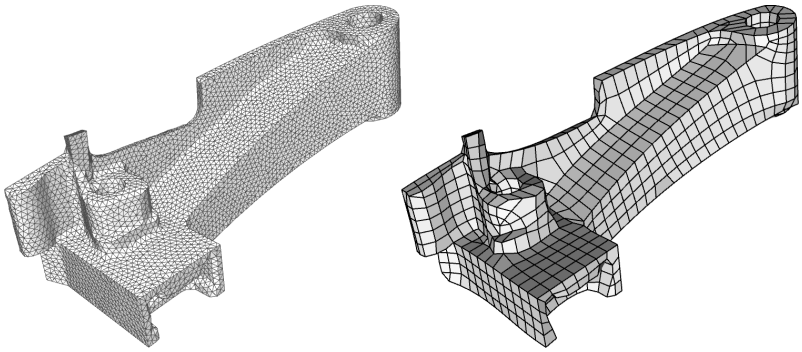
- A curve shall be mapped to a straight line.
- A closed curve shall be mapped to a circle.
- Certain lines shall be perpendicular or parallel to each other.
- A feature curve that is (approximately) planar shall be preserved.
- A region that is (approximately) planar shall be preserved.

Note that it is not necessary to prescribe the exact position, orientation or size of the features considered, only certain geometric properties.

Constraints like these have been scarcely considered in the literature of mesh parametrization. In this paper, we give an analysis of these constraints and develop algorithms that are capable of enforcing them. We assume that the constraints are predetermined by the user, i.e. we do not consider methods for the detection of features, symmetries, etc. Also, we restrict our scope to the parameterization of surfaces with multiply connected disc topology; thus we do not concern ourselves with problems related to cutting or segmenting closed meshes prior to parameterization.



(a) *Texture mapping* **(b)** *Surface fitting*



(c) *Quad remeshing*

Figure 2: *Examples of parameterization applications.*

We first summarize the necessary mathematical background (Chapter 1), then give an overview of the relevant literature, focusing on methods capable of enforcing geometric constraints (Chapter 2). Next, we discuss how to reduce typical higher-level geometric constraints to low-level relations between positions or angles (Chapter 3). As our main contribution, we present extensions to three widely used parameterization methods: Discrete Natural Conformal Mappings (Chapter 4), Angle-Based Flattening (Chapter 5) and As-Rigid-As-Possible Parameterization (Chapter 6) and also develop a method to iteratively deform an existing parameterization such that it satisfies our constraints (Chapter 7). Finally we discuss the details of our implementation, evaluate and compare the performance of the different methods, and present a practical application of the presented algorithms in the context of reverse engineering (Chapter 8).

Chapter 1

Preliminaries

The results presented here can be found in the literature, e.g. in [Pinkall and Polthier, 1993] and [Hormann et al., 2007], we include them in a succinct form to make our treatment more self-contained.

Triangle meshes, differential geometry A *triangle mesh* M is given as a set of vertex positions in $\mathbb{R}^3$: $p_i = (x, y, z), i = 1, \ldots, N_V$, together with a simplicial complex (V, E, F) describing its topology. A *parameterization* of the mesh is a mapping of M to the Euclidean plane - $f : (x, y, z) \longmapsto (u, v)$.

We often consider the dual mesh, which is constructed[1] by replacing each face with a *dual vertex* in its circumcenter, each edge with a perpendicular *dual edge*, and each vertex with a *dual cell*.

For a general surface in $\mathbb{R}^3$, we can measure its curvature at a given point, by intersecting it with a plane parallel to the normal vector. The intersection is a plane curve that has a well defined curvature at the given point, called the *normal curvature*. Its maximal and minimal values (always attained in orthogonal directions) are called the *principal curvatures* at the point. The product of the principal curvatures is the *Gaussian curvature*.

It is a fundamental result of differential geometry that the Gaussian curvature is an *intrinsic* property of the surface, invariant to isometric deformations. As the plane has 0 Gaussian curvature everywhere, surfaces that have non-zero curvature cannot be mapped to the plane without distortion. Surfaces with zero Gaussian curvature are called *developable*.

On triangle meshes, we only consider Gaussian curvature at the vertices where it is actually equal to the *angle defect* of the triangle fan, see Figure 1.1

$$K = 2\pi - \sum \alpha.$$

[1] We assume a circumcentric, or Voronoi dual.

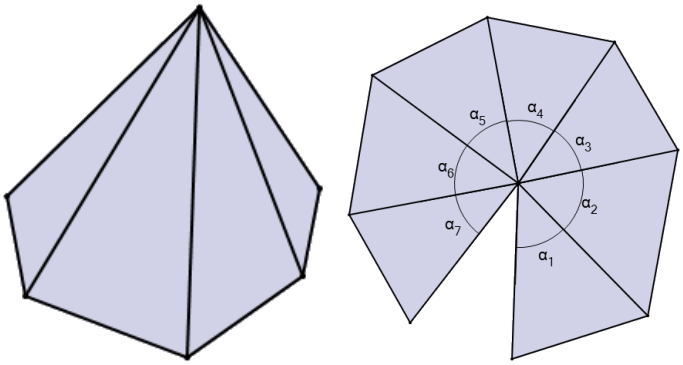


Figure 1.1: *Triangle fan of a vertex (a) in* $R^3, (b)$: *flattenedisometrically.*

Gradient in a triangle An important assumption about a triangle mesh parameterization is that it is *piecewise-affine*, i.e. it is uniquely determined by the images of the vertices. As a consequence, the derivatives of the parameterization are constant within each triangle. Using barycentric coordinates and choosing an arbitrary local coordinate system[2] (X, Y) it is easy to show that if $u_1, u_2, u3$ are the u-coordinates of the vertices of a triangle, the gradient (the vector of first derivatives) of the u coordinate function can be computed, (with the notations of Figure 1.2) as the following:

$$\nabla u = \begin{bmatrix} \frac{\partial u}{\partial X} \\ \frac{\partial u}{\partial Y} \end{bmatrix} = \frac{1}{2A_{p_1,p_2,p_3}} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} e_{23} & e_{31} & e_{12} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} =$$

$$= \frac{1}{2A_{p_1,p_2,p_3}} \begin{bmatrix} Y_3 - Y_2 & Y_2 - Y_1 & Y_1 - Y_3 \\ X_2 - X_3 & X_1 - X_2 & X_3 - X_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix},$$

where A is the area of the triangle.

Using the gradient vectors $\nabla u, \nabla v$ one can form[3] the Jacobian matrix J of the parameterization, which is again constant for each triangle of the mesh:

[2] In practice, we use (after normalization) e_{12} and $N \times e_{12}$, where $N = e_{12} \times -e_{31}$ is the triangle normal.

[3] In the differential geometry literature, along with several surveys on parameterization, the Jacobian matrix is often defined as the transpose of the matrix that we have given. Our convention corresponds to the practically relevant case when vectors are multiplied by matrices from the left.

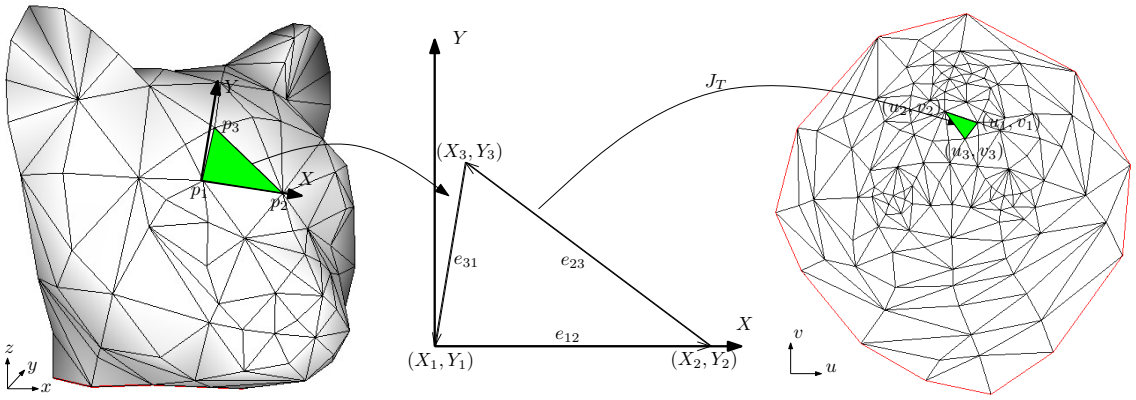


Figure 1.2: *Notations for gradient in triangle.*

$$J = \begin{bmatrix} \frac{\partial u}{\partial X} & \frac{\partial u}{\partial Y} \\ \frac{\partial v}{\partial X} & \frac{\partial v}{\partial Y} \end{bmatrix} = \begin{bmatrix} \nabla u^T \\ \nabla v^T \end{bmatrix} =$$

$$\frac{1}{A_{p_1,p_2,p_3}} \begin{bmatrix} Y_3 - Y_2 & Y_2 - Y_1 & Y_1 - Y_3 \\ X_2 - X_3 & X_1 - X_2 & X_3 - X_1 \end{bmatrix} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \end{bmatrix}.$$

Note that the Jacobian can be interpreted as the linear part of the affine mapping that brings the triangle from a reference position (as per the chosen local coordinate frame), to its place in the parameterization, as illustrated in Figure 1.2.

Conformal mappings, the cotangent Laplacian Most of the common parameterization algorithms make use of the analytical theory of conformal mappings and the so-called cotangent discretization of the Laplacian.

As a starting point, consider Dirichlet's principle, which states that a conformal mapping minimizes Dirichlet's energy [Courant, 1950]:

$$E_D = \frac{1}{2} \int \|\nabla u\|^2 + \|\nabla v\|^2 \, dA. \tag{1.1}$$

It is a well known fact of complex analysis, that conformal mappings are equivalent to complex differentiable functions and thus, they satisfy the Cauchy-Riemann equations [Needham, 1999]:

$$\frac{\partial u}{\partial X} = \frac{\partial v}{\partial Y}$$

$$\frac{\partial u}{\partial Y} = -\frac{\partial v}{\partial X}.$$

As a consequence, conformal mappings are also also harmonic, i.e. their coordinate functions satisfy the Laplace equation:

$$\Delta u = 0$$

$$\Delta v = 0$$

Rewriting the integral as a finite sum over triangles than using the previous expression for the gradient one can easily prove[4] [Pinkall and Polthier, 1993] that on triangle meshes the Laplacian can be approximated by a symmetric, positive definite matrix L, with the following entries (see Figure 1.3):

$$L_{ij} = \begin{cases} -(\cot(\theta_{ij}) + \cot(\theta_{ji})) & i \neq j, e_{ij} \in E \\ \sum_{e_{ik} \in E} (\cot(\theta_{ik}) + \cot(\theta_{ki})) & i = j \end{cases}$$

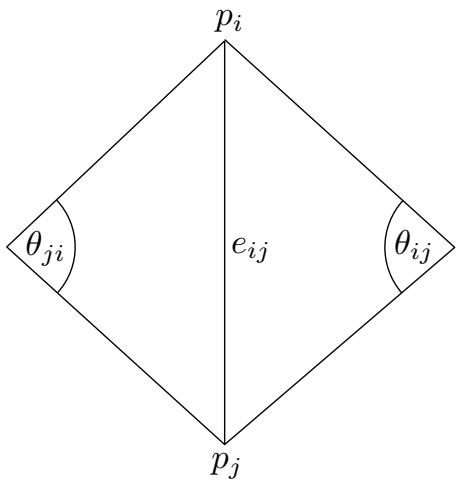


Figure 1.3: *Notations for the cotangent Laplacian.*

Matrices with this structure (diagonals holding the sums of the corresponding rows) are called *Laplacian matrices*[5].

Observe that for certain geometries, the off-diagonal entries of this matrix can become positive, which can lead to artifacts, e.g. triangles flipping their orientation. There are alternative discretizations of the Laplacian on triangle meshes [Floater, 2003], [Belkin et al., 2008], but they lead to matrices that are not symmetric or much less sparse. It is theoretically impossible to construct a 'perfect' discrete Laplacian on general triangle meshes [Wardetzky et al., 2007].

[4] The same expression can be derived using the Finite Element Method or Discrete Exterior Calculus [de Goes et al., 2013].

[5] Note how such a matrix acts similarly to the continuous Laplacian: at a given point, it computes the difference between the value of the function and its local average.

Chapter 2

Overview of previous results

2.1 Parameterization algorithms

Mesh parameterization has an extensive and diverse literature. We refer the unacquainted reader to one of the several surveys and tutorials on the topic [Floater and Hormann, 2005], [Sheffer et al., 2006], [Hormann et al., 2007] or Chapter 5 of the textbook [Botsch et al., 2010]. As we have mentioned in the introduction, we limit our scope to the parameterization of (possibly multiply connected) discs, i.e. we ignore the wide range of methods developed for the *global parameterization* of closed meshes.

The simplest nontrivial methods for mesh parameterization are based on Floater's generalization [Floater, 1997] of Tutte's theorem [Tutte, 1963] characterizing straight edge drawings of planar graphs: [Pinkall and Polthier, 1993], [Eck et al., 1995], [Floater, 2003]. These methods require us to fix the boundary of the mesh on a convex polygon. This limitation can be relaxed by adding some sort of 'scaffolding' to the mesh [Lee et al., 2002], [Kós and Várady, 2003], [Kami et al., 2005], but methods that explicitly optimize the shape of the boundary are considered more practical. These fall into two major categories: conformal methods, aiming for the preservation of angles; and area- or length-preserving methods.

Conformal methods The first published conformal method was Discrete Natural Conformal Paramterization (DNCP) [Desbrun et al., 2003], which - along with the equivalent Least-Squares Conformal Mappings (LSCM) [Lévy et al., 2002] - exploits the analytical properties of conformal mappings and requires the solution of a homogeneous system with a Laplacian matrix. After two vertices have been fixed, these methods compute a mapping that is very close to being exactly angle-preserving with a 'natural' boundary. Still, the quality of the parameterization depends on the choice of the two vertices in a non-trivial way. Spectral Conformal Parameterization (SCP) [Mullen et al., 2008] distributes the constraint on the whole boundary by solving a generalized eigenvalue problem for the Laplacian matrix. This method can also be generalized to polygonal meshes [Alexa and Wardetzky, 2011], [Bouaziz et al., 2012].

Angle-Based Flattening (ABF) [Sheffer and de Sturler, 2001] enforces conformality in a direct way, by minimizing the deviation of the flattened angles from the original ones in a least-squares sense, while satisfying a set of constraints ensuring the planar nature of the result. This leads to a nonlinear, non-convex optimization problem, which can be solved via Newton's method. The method can be made more practical by reducing the size of the Hessian (ABF++) [Sheffer et al., 2005], using a progressive mesh data structure (Hierarchical ABF) [Sheffer et al., 2005] or by linearizing the constraints (LinABF) [Zayer et al., 2007].

More modern methods for conformal parameterization [Kharevych et al., 2006], [Springborn et al., 2008], [Ben-Chen et al., 2008], [Crane et al., 2011], [Myles and Zorin, 2012] follow the Discrete Differential Geometry paradigm [Grinspun et al., 2008], by developing a proper discrete notion of conformality, instead of merely approximating the continuous mathematics. These methods exploit far-reaching mathematical connections between conformality, metric distortion and curvature and optimize for the scaling factors of the edge lengths.

Area-preserving and isometry-preserving methods Conformal methods have very advantageous properties, most importantly their low computational cost; but their lack of regard for area and length distortion could be problematic for many applications. Algorithms that try to strike some balance between angle and area reservation or isometry typically require expensive non-linear, non-convex optimization, thus scaling considerably worse than conformal methods.

In one of the earliest works on parameterization [Maillot et al., 1993] proposed an elasticity-based method, that involves optimization with rational functions, derived from the Green-Lagrange deformation tensor.

The MIPS method [Hormann and Greiner, 2000] computes the 'most isometric' mapping with free boundary by minimizing a nonlinear, transcendent function.

Some methods, stemming from the groundbreaking work of [Sander et al., 2001] iteratively improve some kind of distortion measure for an existing parameterization [Sander et al., 2002], [Sorkine et al., 2002], [Degener et al., 2003], [Dong and Garland, 2007].

All of these methods require expensive nonlinear optimization. This is in sharp contrast with [Liu et al., 2008], which applies the local-global variant of the As-Rigid-As-Possible mesh deformation algorithm [Alexa et al., 2000], [Igarashi et al., 2005], [Sorkine and Alexa, 2007] to parameterization. Although, this method also minimizes a nonlinear energy, measuring the local isometry of the parameterization, it does so by alternating between a local and global optimization phase, which only requires singular value decomposition of 2×2 matrices and a linear solve with a (constant) Laplacian matrix respectively.

In [Li et al., 2010] the parameterization problem for a sheet metal component is solved via physical principles by inverting the metal forming procedure using the so-called one-step approach [Guo et al., 2000], that constitutes of a non-linear Finite Element problem.

Preventing and resolving overlaps It should be noted that most of the mentioned parameterization algorithms cannot guarantee that the resulting mapping will be one-to-one, which means that some of the triangles could invert their orientation, or overlap. Because of this a significant part of the relevant literature concentrates on the problem of preventing or resolving such artifacts [Eckstein et al., 2001], [Tang et al., 2003], [Kraevoy et al., 2003], [Kami et al., 2005], [Lee et al., 2008],[Yu et al., 2012].

Almost all of the common parameterization methods employ the famous cotangent discretization of the Laplacian in some shape or form, and the lack of injectivity usually stems form the fact that the cotangent weights can become negative on lower quality (non-Delaunay) triangulations. There are ways to improve the situation without explicitly modifying the original mesh [Fisher et al., 2007], [Mullen et al., 2011], but these are computationally expensive options. A commonly used heuristic approach is that if a triangle happens to flip, add some additional positive weight to the corresponding entries of the Laplacian matrix and recompute the parameterization[1], see [Bommes et al., 2009] for a relatively sophisticated variant of this strategy. Although this heuristic could work very well in practice (see our solution later), there is no guarantee for convergence let alone its rate.

We would also like to mention the recent progress in optimization with explicit (non-convex) injectivity constraints [Bommes et al., 2013] [Schüller et al., 2013] and the work of Lipman on the theory of injective parameterizations of triangle meshes [Lipman, 2012], [Lipman, 2013].

2.2 Geometric constraints

Parameterization We know of only a few, isolated works in the parameterization literature that consider some kinds of geometric constraints. In [Bennis et al., 1991] and [Azariadis and Aspragathos, 2001] the problem of preserving the shape (geodesic curvature) of feature curves is investigated. Mesh parameterization is an essential part of applications for cloth and garment modeling, where it is crucial that the length of certain feature curves are preserved. [Wang, 2008] and [Igarashi et al., 2009] give methods that enforce this constraint. [liang Chen et al., 2011] gives a method for the preservation of shape of feature curves that is very similar to what we have formulated for LSCM, but enforces them as part of a physics-inspired finite element-based algorithm. [Vallet and Lévy, 2009] extends the ABF algorithm to handle line-based geometric constraints by adding simple linear constraints on the flattened angles to the optimization problem, and has been our main inspiration for our approach in Chapter 5.

[1] Personal communication of Kai Hormann.

Constrained modeling and fitting Although geometric constraints are relatively novel to the subfield of mesh parameterization, they are in no way unknown to the wider field of computer geometry. The enforcement of geometric constraints, known as *constrained modeling*, is a quintessential part of any geometric modeling or CAD system and was already considered in some of the pioneering works on computer-aided design [Sutherland, 1964], [Gopin, 1978], [Light and Gossard, 1982]. Constrained modeling has a very voluminous literature, and is a topic ongoing research; we refer to the surveys [Brüderlin and Roller, 1998], [Hoffmann and Joan-Arinyo, 2005], [Jermann et al., 2006] and [Bettig and Hoffmann, 2011]. An important difference between our approach and traditional constrained modeling techniques is that we operate with 'unorganized' triangle meshes, not the 'organized' models made up from larger scale geometric primitives common in CAD-systems. The constrained fitting problem in reverse engineering [Benko et al., 2002] is another related research field.

Constrained mesh deformation More closely related is the actively developing field of constrained mesh deformation [Masuda et al., 2007], [Gal et al., 2009], [Zheng et al., 2011], [Habbecke and Kobbelt, 2012], [Deng et al., 2013] see the recent survey [Mitra et al., 2012]. Our approach is novel in the sense that the features we are considering are not required to be present in the original model. One recent work that is close to ours in spirit is [Bouaziz et al., 2012], where certain kinds of geometric constraint are enforced during the deformation process by a generalization of the local-global algorithm for the minimization of the ARAP energy [Sorkine and Alexa, 2007]: first, for each constrained set of vertices they fit them with the 'best' shape satisfying the constraints independently, then merge these 'projections' together in a global step, and iterate until convergence. We do something similar in our post-processing algorithm, but we use a curvature flow to deform the constrained vertices to the desired shape, if it is a circle and keep their positions fixed throughout the ARAP iterations.

Chapter 3

Geometric Constraints

3.1 Analysis of possible constraints

It is time to turn our attention to the main topic of our paper, which is the enforcement of geometric constraints during the parameterization process. First, let us survey the kind of constraints we will consider:

- A set of vertices shall be mapped to a straight line.
- A set of vertices shall lie on a circle.
- A set of edges shall map to some prescribed shape.
- Two lines or edges should be perpendicular or parallel.
- A curve of a coplanar sequence of edges in 3D shall retain its shape after being mapped to the parameter plane.
- A developable region shall retain its shape after being mapped to the parameter plane.

As we are working with an 'unstructured' triangle mesh, these requirements should be transformed into lower level constraints involving angles, vertices or edges before they can be enforced during the parameterization. This requires nontrivial arguments (independent of the employed parameterization method) only in the case of mapping vertices to a circle, or if a planar curve is to be preserved.

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 a parameterization of the mesh, the related vertices lie on a circle, i.e. the image of the loop is a *cyclic polygon* and each of the edges keeps its length or gets scaled by the same factor. It is a well-known property of cyclic polygons that the side lengths divide the length of the whole polygon as the respective sector angles

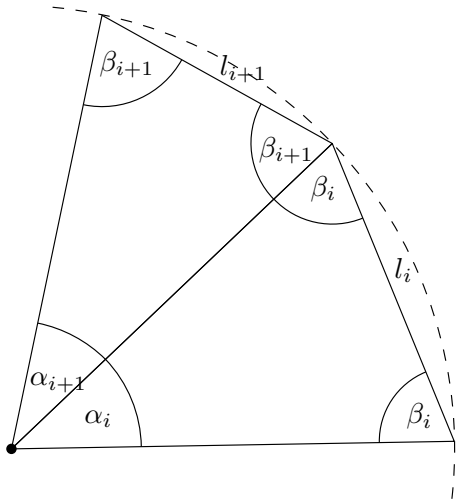


Figure 3.1: *Notations for the circle constraints.*

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 3.1):

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

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

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

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

If the loop is actually an inner boundary curve 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 planar curves, the situation is trivial if the edges are exactly coplanar. In other cases, we fit a plane to the vertices, by taking the eigenvectors corresponding to the two largest eigenvalues to their covariance matrix (after removing their mean), then project the vertices on it for the angle calculations.

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 make a prescribed angle in the parameterization.
- A triangle shall keep its angles and/or edge lengths.

3.2 Evaluation of known parameterization methods

Now that we have our higher-level, 'semantic' constraints to lower-level ones, we can evaluate the known parameterization methods according to their potential for enforcing such constraints. We only consider free-boundary methods.

First, it should be noted that the low-level constraints can be expressed (up to similarity) as linear equations in the positions, thus they can be added as hard or soft constraints to any parameterization method that optimizes for the positions directly. Of these the simplest methods are arguably the angle-preserving DNCP [Desbrun et al., 2003] and the equivalent LSCM [Lévy et al., 2002], which minimize a quadratic form of the vertex positions, thus our constraints can be enforced easily by Lagrange multipliers or adding penalty terms. This is our approach in Chapter 4. Most other methods that optimize for positions require computationally expensive nonlinear optimization or use a per-vertex iterative method, and it is not yet obvious how to enforce our constraints in such a context.

Geometric constraints can also be expressed in terms of mesh angles, which makes the angle-based ABF method [Sheffer and de Sturler, 2001], more precisely its linearized variant called LinABF [Zayer et al., 2007] a very promising alternative. This possibility has been explored before in [Vallet and Lévy, 2009], but we take into account a wider class of constraints in Chapter 5.

These conformal methods give us practically no control over the 'metric' properties, e.g. we cannot preserve edge lengths without adding expensive quadratic equality constraints to the problem. On the other hand, there is a wide class of algorithms, including methods like Curvature Prescription [Ben-Chen et al., 2008], Conformal Equivalence [Springborn et al., 2008] and Incremental Flattening [Myles and Zorin, 2012] where the optimization variables are actually edge length scaling factors, which seemingly opens up the possibility to directly control the metric properties. Unfortunately, after careful examination we have found that these methods have limited capabilities for accommodating geometric constraints. All of these algorithms build upon a discrete analogue of the differential geometric characterization of conformality, i.e. they assume that the edge length scaling factors are not independent for each edge, but computed from a scalar function defined on the vertices and the actual optimization variables are the values of this function. Although [Springborn et al., 2008] seem to employ a variational principle, i.e. minimize a convex energy, unlike other methods this functional does not correspond to some kind of distortion measure, but is actually reverse-engineered to have the unique, conformally equivalent planar mesh as its minimum. To get another perspective, observe that [Ben-Chen et al., 2008] and [Myles and Zorin, 2012] solve a square linear system of full rank, as a rough approximation of the Conformal Equivalence method, and this system is actually a set of linear constraints ensuring the planarity (zero curvature) of the resulting parameterization. As the system is not underdetermined, adding any additional linear constraint would require us to solve it in a least-squares sense which will not result in a valid planar mesh in general. The key observation is that the assumption of this precise sense of conformality, along

with boundary conditions actually define a unique planar mesh [Springborn et al., 2008], i.e. the only degrees of freedom we have are on the boundary[1]: we can prescribe arbitrary edge lengths and angles for boundary edges but doing so for the interior will most probably violate the assumptions behind these methods and lead to unpredictable results. Although finding ways to incorporate geometric constraints into these methods is an interesting potential research topic, it appears that there might be no straightforward way to do so.

A similar obstruction arises when one considers eigenvector-based methods, such as [Mullen et al., 2008] or [Crane et al., 2011]: (smallest) eigenvalue problems are exceptional in the sense that they find the unique solution to a seemingly difficult nonlinear optimization problem, with the practical computational complexity of a simple linear solve. Adding a simple additional constraint forces the problem out of the scope of traditional numerical linear algebra; but future we plan on investigating the practicality of methods such as [Gander et al., 1989] and [Golub et al., 2000] for its solution in the future.

We also observe that the optimization objective of most common parameterization algorithms, such as DNCP/LSCM [Desbrun et al., 2003], [Lévy et al., 2002], ARAP [Liu et al., 2008], Stretch-minimization [Sander et al., 2001] and its variants, MIPS [Hormann and Greiner, 2000] and Green-Lagrange deformation [Maillot et al., 1993], can be reformulated in terms of the singular values of the Jacobian matrix, see the survey [Hormann et al., 2007] for an overview. As the singular values characterize the distortion of the parameterization for each triangle, it might appear that this will allow us to control the metric properties, as we wish. However, this equivalence of objectives is mostly of theoretical significance: we cannot optimize directly for the singular values i.e. the local linear mappings of the triangles, as we have to take the actual geometry and topology of the mesh into account; which leads us back to the original objectives, expressed in terms of positions, angles, etc. The relation between positions and the singular values is highly nonlinear, thus we conjecture that for practical problem sizes it is computationally infeasible to constrain the Jacobian singular values during the parameterization process.

There is one way, however, to get partial control over the Jacobians in a computationally inexpensive way: the minimization of the ARAP energy by local-global iterations as proposed by [Liu et al., 2008]; which is the topic of Chapter 6.

[1] More precisely our degrees of freedom are Möbius transformations - angle-preserving mappings of the plane to itself [Needham, 1999].

Chapter 4

Discrete Natural Conformal Parameterization (DNCP) with geometric constraints

Among the numerous parameterization algorithms available, angle-preserving methods enjoy the greatest popularity, as conformal mappings have surprisingly elegant and convenient mathematical properties and most methods ultimately require the solution of a single linear system. Among these Discrete Natural Conformal Parameterization - and the equivalent Least Squares Conformal Mapping (LSCM) - are arguably the simplest, and have been already implemented as part of popular modeling software, such as Blender3D, Autodesk 3ds Max or Maya and the computer geometry library CGAL [cga,]. As such, it has been our first choice to explore the possibilities for incorporating geometric constraints into mesh parameterization.

In sharp contrast to the smooth case, triangle meshes of disc topology cannot be conformally mapped to the plane in general. The reason for this is obvious: even a single triangle fan can have nonzero angle defect, i.e. curvature, and as such it cannot be flattened in an angle-preserving way. What we can hope to achieve during parameterization then is conformality in some least-squares sense.

4.1 General algorithm

Recalling what we have discussed earlier in Chapter 1, a conformal mapping minimizes Dirichlet's energy E_D:

$$\underset{\text{for } u,v\in\mathbb{R}^{N_V}}{\text{minimize}} \quad \sum_{T\in F} A_T(\|\nabla u\|^2 + \|\nabla v\|^2)$$

As our mapping cannot be expected to be fully conformal, we shall refer to the fact that

Dirichlet's energy is bounded from below by the area of the image, and this lower bound is attained precisely for a conformal mapping:

$$\frac{1}{4}(\|\nabla u\|^2 + \|\nabla v\|^2) \geq \frac{1}{2}(\|\nabla u\| \, \|\nabla v\|) \geq \frac{1}{2}(\nabla u \times \nabla v) = A$$

Thus it is natural to consider the difference between Dirichlet's energy and the Area functional A as a measure of conformality, called the conformal energy E_C:

$$E_C = E_D - A.$$

We have shown earlier that on a mesh Dirichlet's energy is a quadratic form in the vertex positions with a cotangent Laplacian as its matrix, while the area can be calculated as:

$$A = \sum_{e_ij \in E} u_i v_j - u_j v_i.$$

One can observe that for edges on the interior, the contributions of neighboring faces cancel each other out, leaving those corresponding to boundary edges the only nonzero entries in the matrix.

Unlike the cotangent Laplacian, this quadratic form is not symmetric, with which we deal in the standard way: $A' = \frac{A+A^T}{2}$. The difference of the Laplacian L_D and the area functional A is a symmetric, positive-definite form, denoted by L_C, with the following entries:

$$(L_C)_{ij} = \begin{cases} -(\cot(\theta_{ij}) + \cot(\theta_{ji})) & i \neq j; i,j < N_V \text{ or } i,j \geq N_V \\ \sum_{k \in N_i}(\cot(\theta_{ik}) + \cot(\theta_{ki})) & i = j; i,j < N_V \text{ or } i,j \geq N_V \\ 1 & i < N_V; j \geq N_V; e_{p_i,p_j} \in \partial M \\ -1 & i \geq N_V; j < N_V; e_{p_i,p_j} \in \partial M \end{cases}$$

After setting its derivative to zero, we can minimize this energy by solving the resulting homogeneous linear system. Note that the area functional establishes a coupling between the u and v coordinates, so the system is actually of size $2V \times 2V$. It can be shown that L_C has rank $2V - 4$ [Lévy et al., 2002], which is also obvious from the context: the flattened mesh is defined up to its angles, and has four degrees of freedom: position in the plane, orientation and rotation. So, two vertices have to be fixed to make the system full rank.

Unfortunately, picking the vertices to be fixed is not as trivial a matter as it might appear at first: the resulting parameterization depends in a nontrivial way on this choice. The most natural candidates are the pair of vertices furthest apart on the mesh. For the computation of geodesic distances we have implemented the recently published Geodesics

In Heat method [Crane et al., 2013b], which, by utilizing a heat diffusion analogy only requires the factorization of two (cotangent) Laplacian matrices for the entire mesh, after that the distance field from any subset of vertices can be computed by back-substitution, which has negligible cost relative to the factorization (assuming that we solve the systems with sparse direct methods).

An alternative is Spectral Conformal Parameterization (SCP) [Mullen et al., 2008], which distributes the constraint over the entire boundary - without fixing a single vertex - by solving a (generalized) smallest eigenvalue problem with the matrix L_C. As we have discussed in Chapter 2, adding constraints to eigenvalue problems is highly nontrivial, so we did not pursue this avenue as of yet.

We do implement, however, the area weighting scheme proposed in [Mullen et al., 2008], i.e. we divide each cotangent weight and area term by the area of the corresponding triangle. This is known to lead to smaller area distortion and more natural behavior for meshes with inhomogeneous sampling.

The equivalent[1] LSCM method [Lévy et al., 2002] provides an alternative interpretation of the conformal energy as the least squares error of the *Cauchy-Riemann equations*:

$$E_C = \sum_{T \in F} \left\| \nabla v(T) - \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \nabla u(T) \right\|^2 .$$

4.2 Geometric constraints

As we are minimizing a symmetric, positive definite quadratic form of the vertex positions, we can easily impose linear constraints on the problem:

$$\begin{aligned} \underset{\text{for } u,v \in \mathbb{R}^{N_V}}{\text{minimize}} \quad & \begin{bmatrix} u^T v^T \end{bmatrix} L_C \begin{bmatrix} u \\ v \end{bmatrix} \\ \text{subject to} \quad & C \begin{bmatrix} u \\ v \end{bmatrix} = d \end{aligned}$$

Problems like this are readily solved by the method of Lagrange multipliers, i.e. by forming the Lagrangian dual function and setting its gradient w.r.t. to the original variables to zero, which together with the original constraints forms a square, symmetric, yet indefinite linear system [Boyd and Vandenberghe, 2004]:

$$\begin{bmatrix} L_C & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ \nu \end{bmatrix} = \begin{bmatrix} b_u \\ b_v \\ d \end{bmatrix},$$

[1] The oft-cited original equivalence proof of [Cohen-Steiner and Desbrun, 2002] is incorrect, as pointed out on the website of the author of LSCM. We note that in his comments, the author falsely claims that the DNCP matrix is not symmetric.

where ν is the vector of Lagrange multipliers, and b_u, b_v are the boundary conditions for the u, and v coordinates.

Fortunately, most geometric constraints can be expressed as linear equations in the positions:

- When we require that two edges (whether they share vertex in the mesh or not) shall make an angle in the parameterization, this is equivalent to demanding that one edge is the scaled and rotated version of the other. The scaling can be arbitrary but the most natural choice is the ratio of the edge lengths, which results in curve similar to their original counterparts on the 3D mesh. Quantitatively this can be expressed as:

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

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

- In many practical cases DNCP computes parameterizations that have very small angle distortion, but it is effectively a least-squares method, i.e. it favors nonzero but negligible-after-squared errors over exactly zero ones. So it can happen, especially with relatively coarse, highly curved meshes, that developable regions get distorted to mitigate the distortion of other parts of the mesh. In optimization such problems are traditionally handled via weighting or sparsity-inducing methods such as $\ell 1$-norm minimization, but in our case it can be avoided completely, by requiring that certain regions get mapped in an exactly conformal way, or in other words by a similarity. Recall that a conformal mapping of a triangle satisfies the Cauchy-Riemann equation, i.e. the gradient of the v-coordinate function equals the gradient of the u-coordinate rotated by 90 degrees:

$$\nabla v = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \nabla u$$

  Using the expression of the parameterization gradients presented earlier, this can be written as:

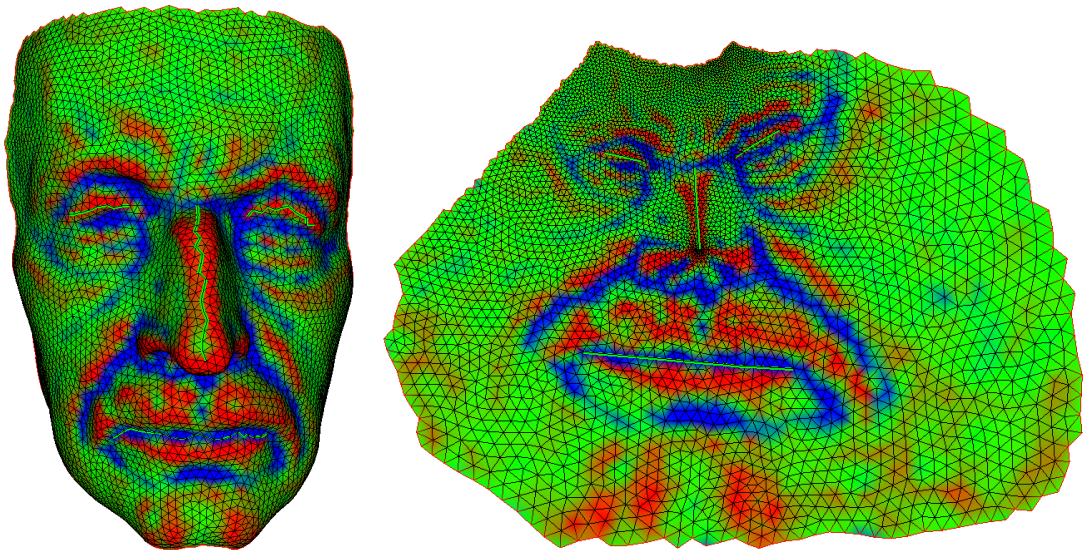
$$(Y_j - Y_k)v_i + (Y_k - Y_i)v_j + (Y_i - Y_j)v_k - (X_j - X_k)u_i - (X_k - X_i)u_j - (X_i - X_j)u_k = 0$$

$$(X_k - X_j)v_i + (X_i - X_k)v_j + (X_j - X_i)v_k - (Y_j - Y_k)u_i - (Y_k - Y_i)u_j - (Y_i - Y_j)u_k = 0$$

4.3 Results

We have found that constraints involving edges on the interior can be enforced without much difficulty, see Figure 4.1 for an example with edges constrained to straight lines. We have run into problems however, when constraining boundary edges: the algorithm can occasionally work if the entire boundary curve is constrained, but if e.g. we want to

preserve only a certain part of it, the parameterization often degenerates. We conjecture that this is due to the employed boundary conditions. We have experimented with several modifications to the boundary conditions, but have not managed to find a universal solution to the problem. Thus, although we plan on investigating this topic further in the future, for now we conclude that this method is only suitable for the enforcement of constraints in the interior of the mesh. The method is capable constraining developable regions to keep their shape up to similarity, but due to the very low angular distortion of the method, the results are visually indistinguishable from the unconstrained ones.



(a) *Model with constraints* **(b)** *Constrained DNCP parameterization*

Figure 4.1: *Results for Maxface model.*

Chapter 5

LinABF with Geometric Constraints

5.1 General algorithm

ABF considers angle-preservation as a direct optimization problem [Sheffer and de Sturler, 2001]. That is, if α_i^* are the original angles and α_i are the angles of the flattened mesh, conformality is equivalent to the problem:

$$\underset{\text{for } \alpha_i^*,\ i=1,\ldots,3T}{\text{minimize}} \quad \sum_{i=1}^{3T} (\alpha_i^* - \alpha_i)^2$$

As most meshes are non-developable, i.e. have non-zero curvature and thus angular defect, we first scale the original angles to add up to 2π around each interior vertex:

$$\alpha_i^* = 2\pi \frac{\alpha_i}{\sum_{(j)} \alpha_j}$$

Unfortunately, this will not result in a valid planar triangulation, thus additional constraints must be imposed on the problem:

- Around each interior vertex the angles add up to 2π.
- In each triangle, the angles add up to π

These are linear constraints on the optimization variables, i.e. we shall minimize a quadratic function with equality constraints, which can be easily done by solving a single linear system. Unfortunately, we must also ensure the consistency of the edge lengths, to avoid situations like the one on the left side of Figure 5.1. This can be done by prescribing the equality of the first and the last edge as we traverse the faces adjacent to an inner vertex. It follows from basic trigonometry, that this is equivalent to the following non-linear, non-convex constraint for each interior vertex:

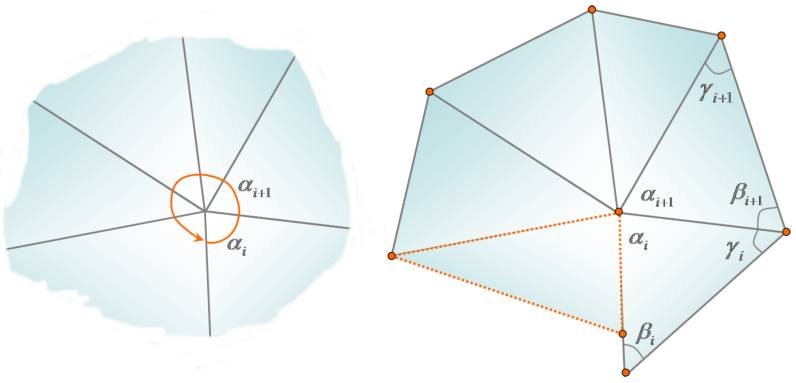


Figure 5.1: *Notations for Angle-Based Flattening. Taken from [Zayer et al., 2007].*

$$\prod_{i \in T_i} \frac{\sin(\beta_i)}{\sin(\gamma_i)} = 1$$

With this constraint, our task becomes a non-convex optimization problem, which can potentially have multiple local minima, and thus might require computationally expensive iterative methods to solve. One could use algebraic manipulations or decimation to reduce the computational cost of iterations [Sheffer et al., 2005], but somewhat surprisingly, a simple linearization of the most difficult constraint leads to a drastic decrease in the required computational effort, with practically negligible impact on accuracy [Zayer et al., 2007].

First, reformulate the problem and the constraints in terms of the (relative) estimation error $e_\alpha = \alpha^* - \alpha$ and take the logarithm of the last constraint:

$$\begin{array}{ll} \underset{\text{for } e_1,\ldots,e_{3T}}{\text{minimize}} & \sum_{i=1}^{3T} \frac{e_i^2}{\alpha_i^{*2}} \\ \text{subject to} & \sum_{i=1}^{N} e_i = 2\pi - \sum_{i=1}^{N} \alpha_i, \forall p \in V \\ & e_\alpha + e_\beta + e_\gamma = \pi - (\alpha + \beta + \gamma), \forall T \in F \\ & \sum_{i=1}^{N} (\log(\sin(\beta_i + e_{\beta_i})) - \log(\sin(\gamma_i + e_{\gamma_i}))) = 0, \forall p \in V \end{array}$$

Finally, take the first-order Taylor-series approximation of the terms, to arrive at an expression, that is linear in the unknowns:

$$\sum_{i=1}^{N} (\cot(\beta_i) e_{\beta_i} - \cot(\gamma_i) e_{\gamma_i}) = \sum_{i=1}^{N} (\log(\sin(\gamma_i)) - (\log(\sin(\beta_i))$$

In conclusion we have to minimize a quadratic function, subjected to linear equality constraints, which could be readily solved by the method of Lagrange multipliers. To simplify the process even further, one could make yet another change of variables: $r_i = \frac{e_i}{\alpha_i}$, which transforms the problem into finding the least-norm solution of an underdetermined linear system:

$$\begin{aligned} \underset{\text{for } r_1,\ldots,r_{3T}}{\text{minimize}} \quad & \|r\|_2^2 \\ \text{subject to} \quad & A \operatorname{diag}(\alpha) = b \end{aligned}$$

This is a standard problem in optimization theory, with a closed-form solution [Boyd and Vandenberghe, 2004]:

$$r^* = C^{\mathrm{T}}(CC^{\mathrm{T}})^{-1}b$$

where $C = A \operatorname{diag}(\alpha)$.

5.1.1 Constructing the mesh from the angles

Having the optimal angles at our disposal, we can construct the planar triangulation by any means suitable. One could, following the original algorithm on ABF, use a trivial greedy algorithm, reconstructing triangles one-by-one, but this method is numerically unstable for larger data sizes. A much more robust alternative - first proposed in [Sheffer et al., 2005] - is to solve a DNCP system (with the matrix entries computed using the flattened angles) for the positions.

As mentioned in Chapter 4, this method requires us to fix two points on the mesh, to set the orientation and scale of the parameterization. As, at this point our mesh is a planar triangulation, the choice shall have no effect on the parameterization outcome.

5.2 Geometric constraints

As we optimize directly for the angles, and our equations form an underdetermined system any geometric constraint that can be defined in such terms can be enforced by modifying existing equations or adding new ones:

- If two edges, connected by a common vertex, are required to make an angle, for interior vertices we split the corresponding equation to two parts, requiring a subset of angles to add up to the required amount of one side and 2π minus the required amount the other side. For boundary edges we simply add a new equation in a similar vein.
- If two edges are required to make an angle but they are not adjacent, we build a path out of triangles between two adjacent faces and add a new equation to the system, requiring that the sum of the corresponding (signed) angles add up to the required amount. See Figure 5.2

- If a triangle is to keep its shape, we simply remove its angles from the optimization problem. Vertices on the boundary of the constrained regions require special care: there the rescaling of at the beginning is done only using the unconstrained angles.

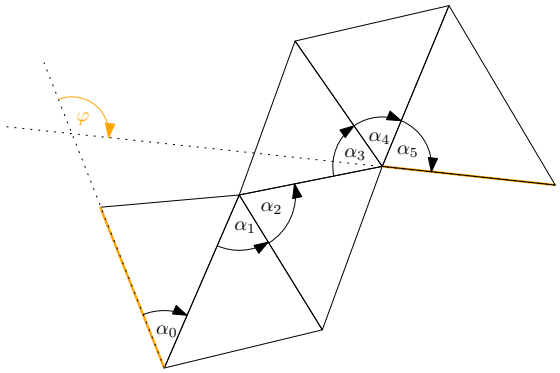
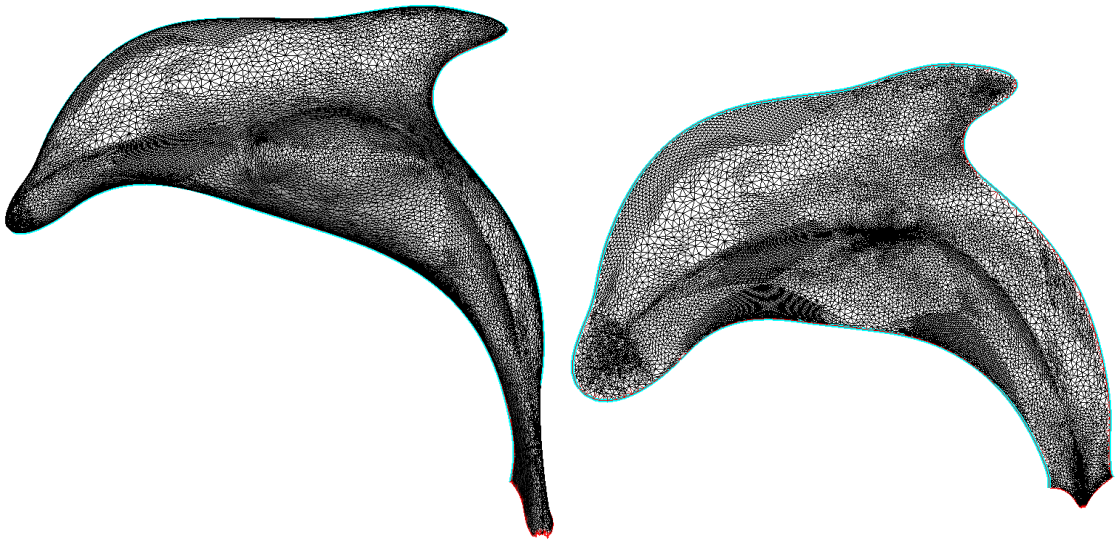


Figure 5.2: *Notations for constraints involving two separate edges in ABF.*

Note that we have roughly $4N_V$ constraints for $6N_V$ unknowns by default, which means that we can add as much as $2N_V$ (non-contradictory) additional constraints to the problem before it becomes overdetermined.

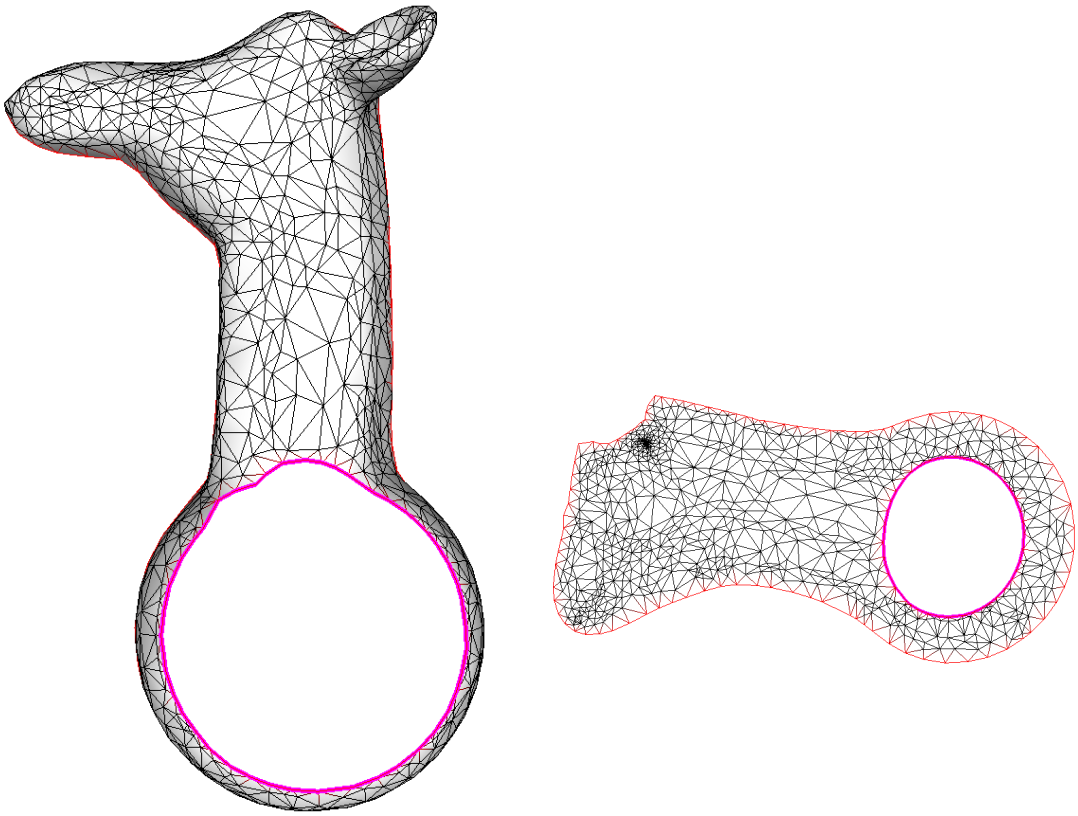
5.3 Results

Some results with the use of this algorithm can be seen in Figure 5.3 and Figure 5.4. An important property of conformal methods can be observed, as it might appear that the algorithm has failed to preserve the exact shape of the curve, but a closer inspection reveals that while the angles have been preserved, the length of the edges are different. It is known, for example, that an N-sided polygon has $N-4$ degrees of freedom, in addition to the N we can constraint via the angles. As the method based on DNCP is incapable of constraining boundary curves, we conclude that the angle-preserving methods we have implemented have limited capabilities to preserve the shape of a planar curve. However, ABF can handle line constraints on both the boundary and the interior (results omitted due to spatial limitations). Similarly to DNCP, the method has very low angular distortion by default, thus constraining developable regions to retain their shape up to similarity has a mostly negligible effect.



(a) *Model with constraints* **(b)** *Constrained ABF parameterization*

Figure 5.3: *Results for Dolphin model. Note the distortion of the boundary.*



(a) *Model with constraints* **(b)** *Constrained ABF parameterization*

Figure 5.4: *Results for Giraffe model.*

Chapter 6

ARAP Parameterization with Geometric Constraints

Angle-preserving parameterization methods are very attractive and widely used, due to their elegant mathematical properties and low computational cost. However area or length preservation can be important in many potential applications, and from the viewpoint of enforcing geometric constraints, it can be said that conformal methods give us little to no control over the metric properties of the parameterization. For example, with LSCM or ABF we can easily preserve the shape of two identical regions of the mesh, but we get no say regarding their size in the parameterization. This limitation motivates our search for a method which optimizes directly fro the metrical properties of the parameterization (i.e. aims at maximal isometry in some sense) and allows for the incorporation of a wider range of geometric constraints. As we have already discussed in, there are many isometric parameterization methods available, but most of them require computationally expensive nonlinear optimization, and can be applied to data sizes typical in current practice only by using sophisticated 'multigrid' linear solvers. There is one known exception: the minimization of the As-Rigid-As-Possible energy through local-global iterations.

6.1 The ARAP energy

Restricted to a single triangle a paramterization acts as an affine mapping, the linear part of which is the Jacobian matrix J, known to be constant within each triangle and represented in a local coordinate system by a two-by-two matrix as see in Chapter. The As-Rigid-As-Possible (ARAP) energy measures the 'distance' (in Frobenius norm) between the Jacobian and the closest rigid transformation (rotation):

$$E_{ARAP}(T) = min_{R \in SO(2)}(\|J - R\|_{\mathrm{F}})$$

To compute the 'most isometric' mapping of the mesh, we can minimize the area-weighted sum of the ARAP energy in each face:

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

As the closest rotations are not known a priori, but depend on the Js in a nontrivial way, this is a nonlinear energy. Its gradient and Hessian has a closed form expression[1], and thus Newton's method can be applied to its minimization [Chao et al., 2010], but instead we consider a simpler, computationally less expensive alternating optimization algorithm, known in the literature as the Local-Global method [Sorkine and Alexa, 2007], [Liu et al., 2008].

The key observation of the Local-Global algorithm is that, if we fix either one of the terms in the per-triangle energy, we get a very simple problem:

- Given a mapping and thus its Jacobian, we want to find the rotation matrix closest to it. This can be computed via Singular Value Decomposition (SVD) or Polar Decomposition of a 2×2 matrix. This can be done independently for each triangle, thus this is called the *local phase*.
- Given a set of rotations we want to compute a mapping, represented by its Jacobians, that fits them in a Least-Squares manner. This requires the solution of a sparse linear system for the vertex positions, so it is called the *global phase*.

It is easy to see that neither of these steps can increase the energy, so by alternating between them, we can find a local minimum.

The ARAP energy was first considered for 2D and 3D mesh deformation [Alexa et al., 2000], [Igarashi et al., 2005], [Igarashi et al., 2009], [Sorkine and Alexa, 2007]. Its application to parameterization is due to [Liu et al., 2008]. Although the local-global algorithm could appear an ad-hoc procedure, it was noted in [Bouaziz et al., 2012] that it belongs to a well-known class of large-scale nonlinear optimization methods known as proximal algorithms [Parikh and Boyd, 2013].

6.1.1 Local phase

Finding the optimal rotation aligning two corresponding sets of points is known as the Orthogonal Procrustes Problem [Gower and Dijksterhuis, 2004], which has a solution using the SVD of the covariance matrix of the positions, or in our case, the Jacobian.

[1] They are very similar to the cotangent Laplacian.

Indeed, given the SVD of the Jacobian matrix of a triangle:

$$J_T = U_T \Sigma_T V_T^{\mathrm{T}}$$

The optimal rotation is given as:

$$R_T^* = V_T U_T^{\mathrm{T}}$$

For 2×2 matrices, the situation is even simpler, as there is a closed form expression available. First, instead of the optimal rotation we compute the optimal similarity transform:

$$\begin{aligned} &\underset{\text{for } S_T, T \in F}{\text{minimize}} && \textstyle\sum_{T \in F} A_T \left\| J_T - S_T \right\|_F \\ &\text{subject to} && S_T = \begin{bmatrix} a_T & -b_T \\ b_T & a_T \end{bmatrix} \end{aligned}$$

This is a diagonal quadratic form and with $J = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ its minimization results in the following:

$$S_T = \begin{bmatrix} \frac{a+d}{2} & \frac{c-b}{2} \\ \frac{b-c}{2} & \frac{a+d}{2} \end{bmatrix}$$

For a rotation it is required, that $\det(R) = 1$. At this point this can be achieved by simply dividing S, with its determinant:

$$R_T = \frac{S_T}{\det(S_T)}$$

A popular alternative is to compute the Polar Decomposition of J: $J = YR$, where R is by definition the rotation closest to J in Frobenius norm. As it was observed by [Chao et al., 2010], this allows for a mechanical interpretation of the ARAP energy: in the context of finite strain theory J is called the *deformation gradient*, while the term $Y - I$, where I is the identity matrix is known as the *Biot strain* [Biot, 1938].

6.1.2 Global phase

Given a set of rotations our task is to find a parameterization, the Jacobians of which fit the rotations (in a least-squares sense). It can be shown that this is equivalent to minimizing a quadratic form with the cotangent Laplacian, which is in turn requires the solution of a pair of sparse linear systems for the positions [Liu et al., 2008]:

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

$$L_{ij} = \begin{cases} -(\cot(\theta_{ij}) + \cot(\theta_{ji})) & i \neq j \\ \sum_{k \in N_i} (\cot(\theta_{ik}) + \cot(\theta_{ki})) & i = j \end{cases}$$

$$b_{u,i} = \sum_{j \in N_i} \left[\cot(\theta_{ij})(a_{T(ij)}(X_i - X_j) + b_{T(ij)}(Y_i - Y_j)) + \cot(\theta_{ji})(a_{T(ji)}(X_i - X_j) + b_{T(ji)}(Y_i - Y_j)) \right]$$
$$b_{v,i} = \sum_{j \in N_i} \left[\cot(\theta_{ij})(a_{T(ij)}(Y_i - Y_j) - b_{T(ij)}(X_i - X_j)) + \cot(\theta_{ji})(a_{T(ji)}(Y_i - Y_j) - b_{T(ji)}(X_i - X_j)) \right],$$

where $a_{T(ij)}, b_{T(ij)}$, are the elements of the rotation matrix $R_{T(ij)} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$ for triangle $T(ij)$ of e_{ij}, while X_i, Y_i are local coordinates of p_i, as per Chapter 1.

These are actually *Poisson's equations* $\Delta f = \nabla \cdot g$ with the *discrete divergences* of the gradient vector fields on the right hand side.

The coefficient matrix remains constant throughout the iterations, which means that we can factor it once and only do forward and back-in each iteration.

6.1.3 Initialization with rotation field

Up until now, we have ignored the problem of initializing the local-global iterations. The most common approach is to take an arbitrary mapping as our starting point and proceed with a local step. This approach has its merits for many potential applications, and the choice of the concrete mapping has surprisingly small effect on the speed of convergence [Liu et al., 2008], but can be considered ill-suited if we would like to incorporate geometric constraints. Thus, although some specific problems involving geometric constraints might benefit from this 'global' initialization, we have instead decided to fix the rotations as the first step of our algorithm. This approach has been first considered by Myles and Zorin [Myles and Zorin, 2012] in the context of quad mesh generation on closed meshes and it is inspired by state-of-the-art methods for vector field generation.

The main idea stems from the observation that our rotations need not define a valid parameterization of the mesh, they should only be not too far from one. Let us see how one would approach constructing such a rotation field: first, fix an arbitrary triangle of the mesh on the plane; then, proceed to recursively flatten its neighbors isometrically, by traversing a spanning tree of faces (or equivalently, dual edges). It is easy to see that this procedure will eventually produce a degenerate result; as on meshes Gaussian curvature at a vertex equals the angular defect, i.e. the amount by which the triangle fan around the vertex will fail to close when flattened isometrically. To get a more fitting set of rotations then, we shall apply some amount of additional rotation ω_{ij} when traversing a dual edge e_{ij}. The question is: how much additional rotation shall be applied along each dual edge? A natural requirement is that they should make each triangle fan approximately closed,

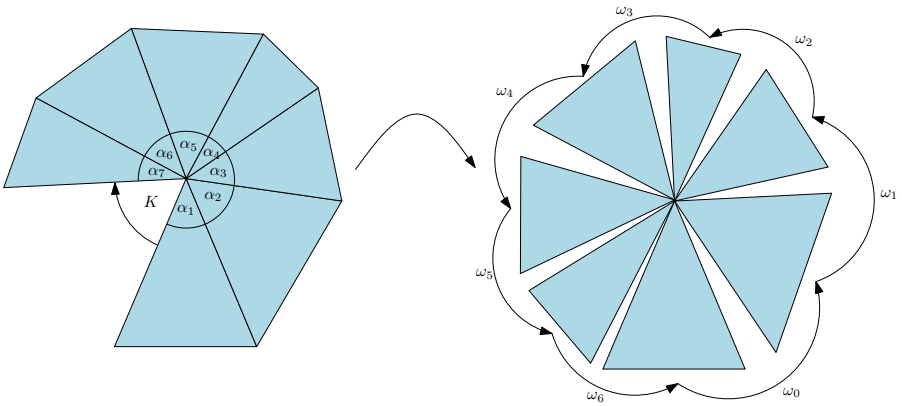


Figure 6.1: *Interpretation of the ARAP initialization algorithm.*

i.e. they should counteract the effect of Gaussian curvature at each of the vertices, see Figure 6.1.

So, for each interior vertex p_i we would like to satisfy the following constraint:

$$\sum_{e_{ij} \in E} \omega_{ij} = 2\pi - \sum \alpha_i.$$

These equations form a sparse, underdetermined linear system, which is guaranteed to have full row rank [Crane et al., 2010] and which we solve in a least-norm manner, to get a set of rotations closest to the ones from naive isometric flattening:

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

where d_0 is the vertex-edge adjacency matrix (the exterior derivative for 0-forms, see [de Goes et al., 2013]), ω is the vector of unknown rotation values for each (dual) edge, and K is the vector of Gaussian curvatures (for the interior vertices).

As before, we solve this problem in the standard way:

$$\omega^* = -d_0^{\mathrm{T}} (d_0 d_0^{\mathrm{T}})^{-1} K.$$

Given these rotation increments, we can construct the rotation set as before by placing a triangle on the plane and traversing a spanning tree of faces, now applying the right amount rotation between each adjacent face.

Although this argumentation might appear ad-hoc, it can be put on firm mathematical grounds. The columns of the rotation matrices are pairs of orthogonal vectors, defined in the local coordinate system of the triangles. Our task then can be interpreted as computing a maximally smooth discrete vector field on the surface, which is an active field of research

the geometry processing. Indeed, the algorithm we have described has appeared several times in the literature on vector field design, in different forms [Kälberer et al., 2007], [Ray et al., 2008], [Bommes et al., 2009], [Crane et al., 2010]. Our approach is practically equivalent to that described in [Crane et al., 2010].

6.2 Geometric constraints

As we employ a multi-phase iterative process, constraints must be enforced during each phase and between iterations. We note beforehand that the shape and alignment of the constrained regions will become fixed during the initialization phase, in the global and local phases we can only control their scale, or can decide to ignore them.

6.2.1 Initialization Phase

As in the case of ABF, we have an underdetermined linear system, to which 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. 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 inner loops in the presence of a user-defined one.

- When two separate edges are required to make an 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 6.3 assume that we have two (oriented) edges e_1 and e_2 constrained to make 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 $\alpha^{'}$ 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 be tempting to do the same for the edges on the boundary of the constrained region, but we have run into numerical stability problems doing so.

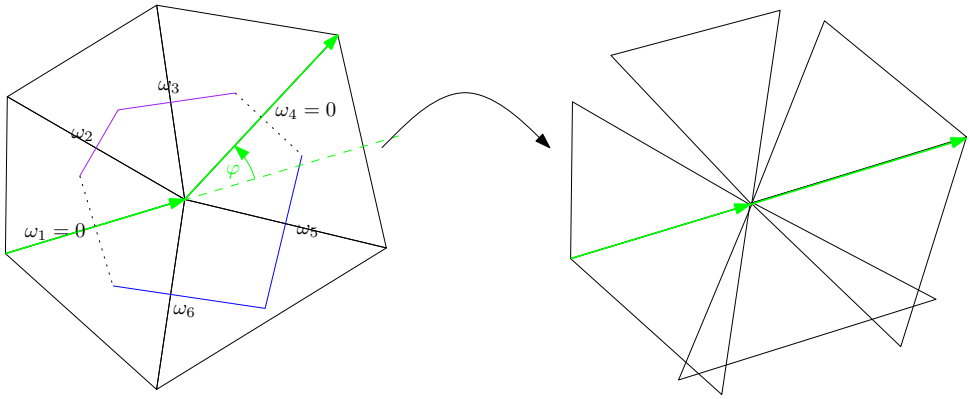


Figure 6.2: *Notations for constraints involving two adjacent edges in ARAP. On the right we illustrate the effect of the constraint, if the prescribed angle is $\varphi = 0$.*

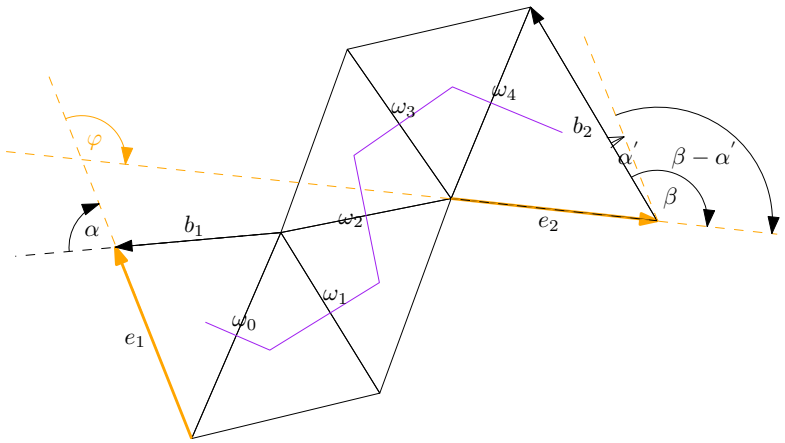


Figure 6.3: *Notations for constraints involving two separate edges in ARAP.*

6.2.2 Global Phase

As for LSCM in Chapter, we enforce linear constraints by the use of Lagrange multipliers so the optimization problem for the u (resp. v) coordinates

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

is solved via the symmetric, indefinite linear system

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

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. This is one of the major advantages the ARAP-based algorithm has over the previous ones: we have control over the absolute scale of the features in the parameterization, while for conformal methods the most we could influence is the relative size of certain, coupled features. Thus, the user has the opportunity to prescribe arbitrary scaling factors for each feature independently. This approach is not without its drawbacks however: examine the task shown in Figure 6.5. One part of the boundary (colored cyan) is to be preserved exactly in the parameterization, while another one (colored green) is to be mapped to a straight line. It is obvious, that it is impossible to satisfy both of these constraints by mapping the boundary edges isometrically. Although in this case we could come up with some tailor-made heuristic to find appropriate scaling factors, the general solution is to allow constraints that enforce geometric properties only up to similarity as done in the case of DNCP. We could still run into problems, if we also prescribe angles between features, see our discussion in Chapter 8.

6.2.3 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.

6.3 Preventing overlaps

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 with geometric constraints, an entire region could 'spill over' a highly curved, concave boundary. We have already discussed methods for the handling of such artifacts in , for our experiments we have implemented a fairly simple heuristic, with great success. If, at the end of a global iteration, 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. We iterate until there are no flipped triangles, or a threshold on the number of iterations has been reached. Although this is a completely ad-hoc procedure, without any theoretical guarantees, it works remarkably

well in practice, and with its help, we have managed to eradicate overlaps completely from all of our examples.

We have also experimented with the more refined weighting scheme in [Bommes et al., 2009], called *local stiffening*. We have found that this variant might be well-suited for the prevention of local, isolated overlaps; but it fails to converge in a reasonable time for the more expansive 'spills' that we have encountered.

6.4 Results

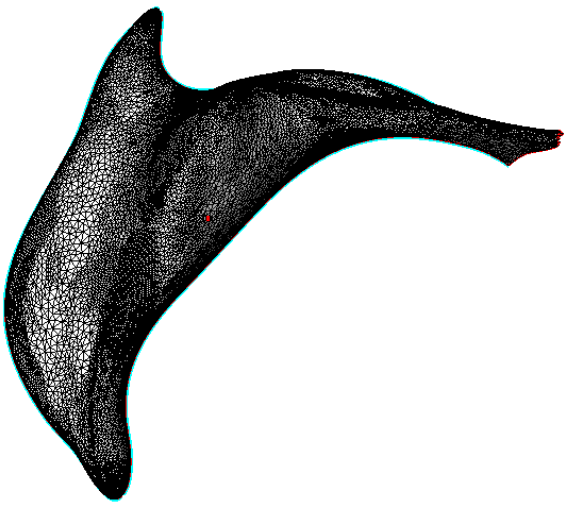
Dolphin model We require one part of the boundary (teal) to preserve its shape. Compared to ABF, with ARAP we can preserve the shape *exactly*, due to the fact that we can keep the length of the edges in addition to the angles. The results with and without constraints are compared in Figure 6.4.

Ford model We require one part of the boundary (teal) to preserve its shape, while the other part (green) to map to a straight line. As we cannot satisfy both of these requirements with isometric boundaries, we preserve the shape of the teal part only up to similarity. The results with and without constraints are compared in Figure 6.5.

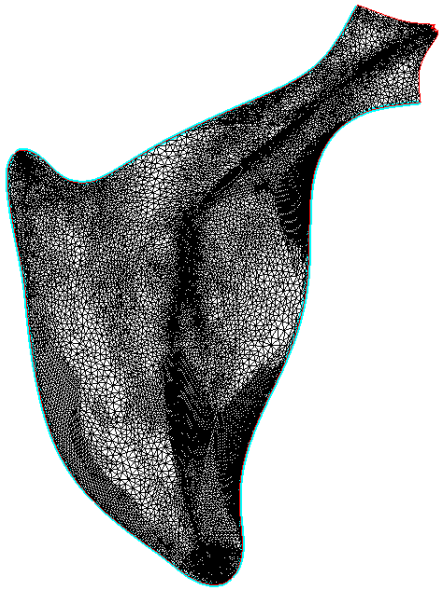
Giraffe model We require one part of the boundary (teal) to preserve its shape, while the other part (purple) to map to a circle. The results with and without constraints are compared in Figure 6.6.

Maxface model We require parts of the boundary (green) to map straight lines, while also constraining them to be perpendicular. The results with and without constraints are compared in Figure 6.7.

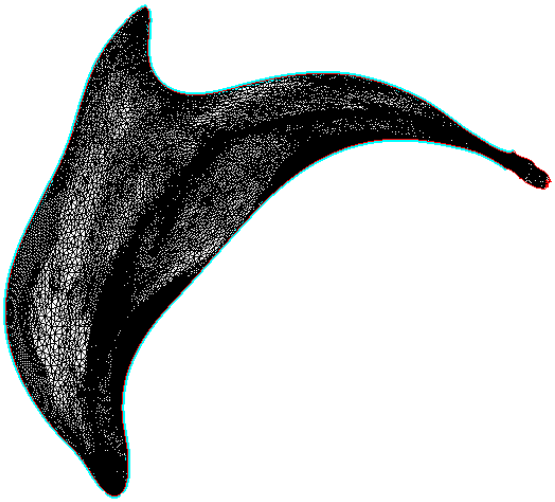
Darmstadt model We require selected regions (green) to keep their shape. The results with and without constraints are compared in Figure 6.8.



(a) *Model with constraints*

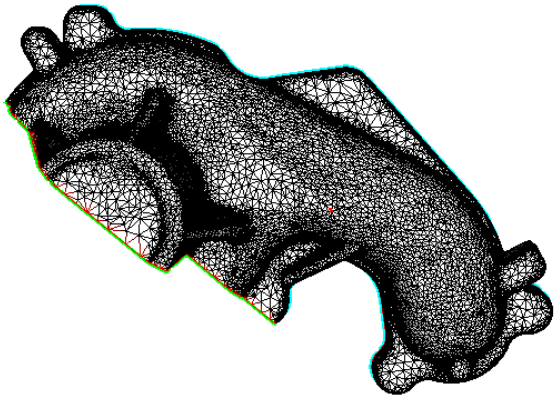


(b) *Unconstrained ARAP parameterization*

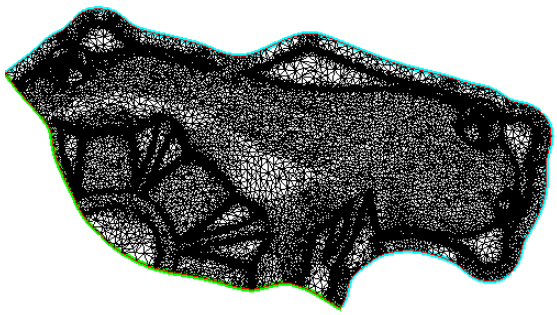


(c) *Constrained ARAP parameterization*

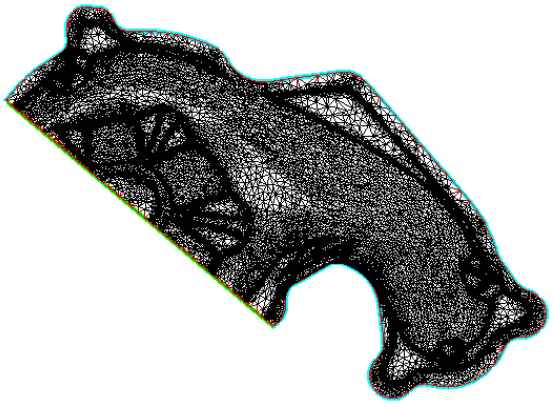
Figure 6.4: *Results for Dolphin model.*



(a) *Model with constraints*

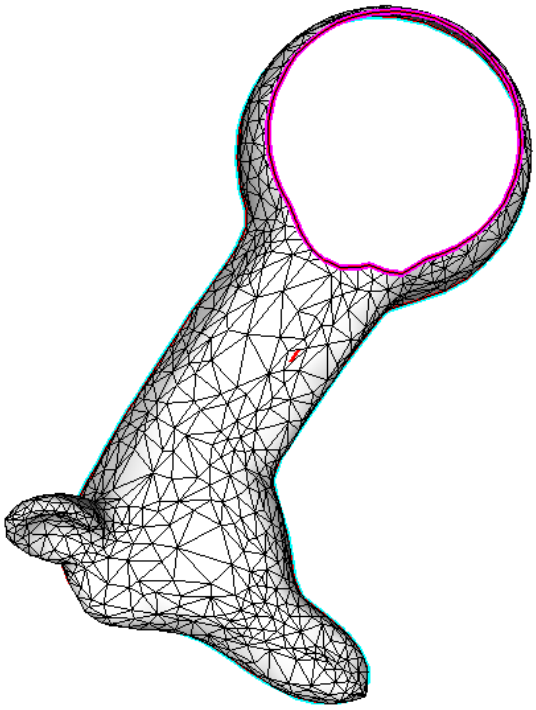


(b) *Unconstrained ARAP parameterization*

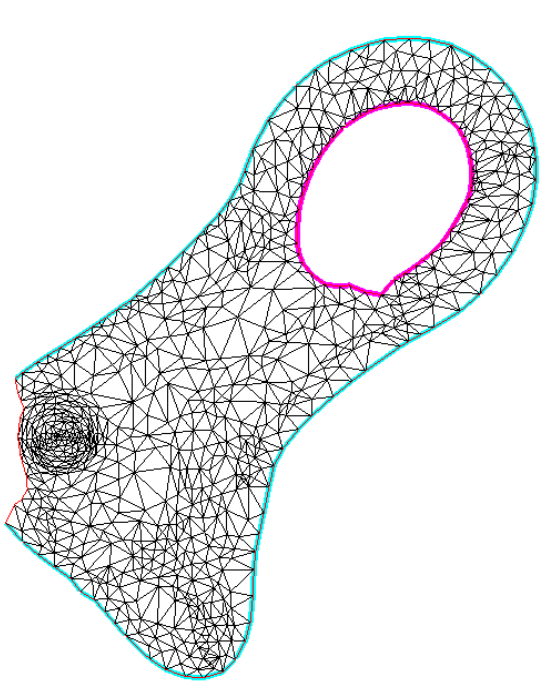


(c) *Constrained ARAP parameterization*

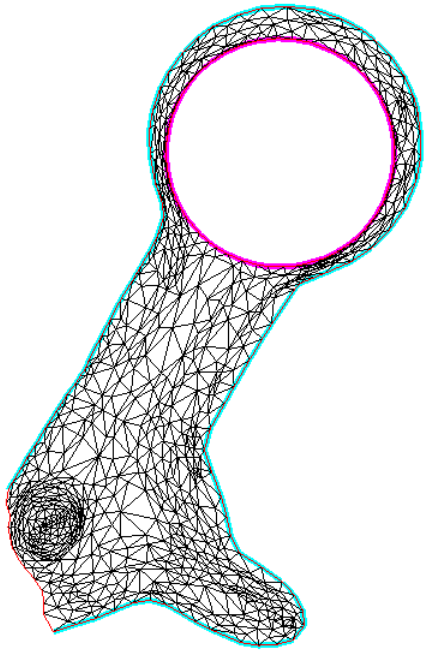
Figure 6.5: *Results for Ford model.*



(a) *Model with constraints*

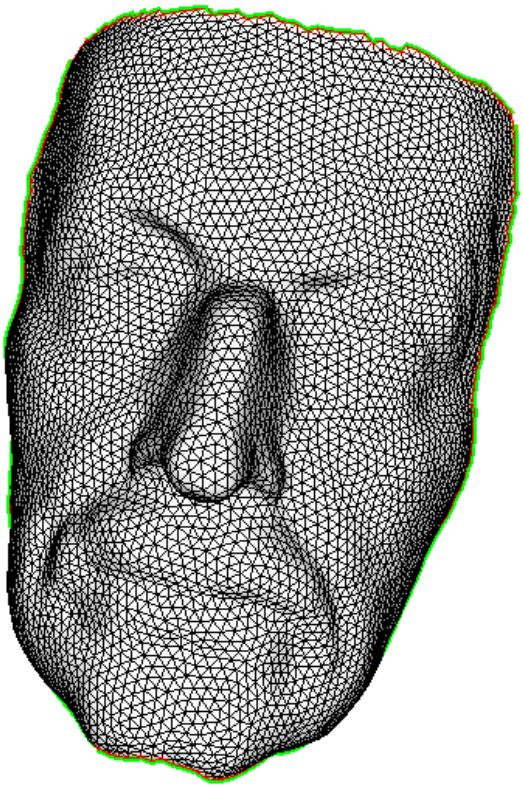


(b) *Unconstrained ARAP parameterization*

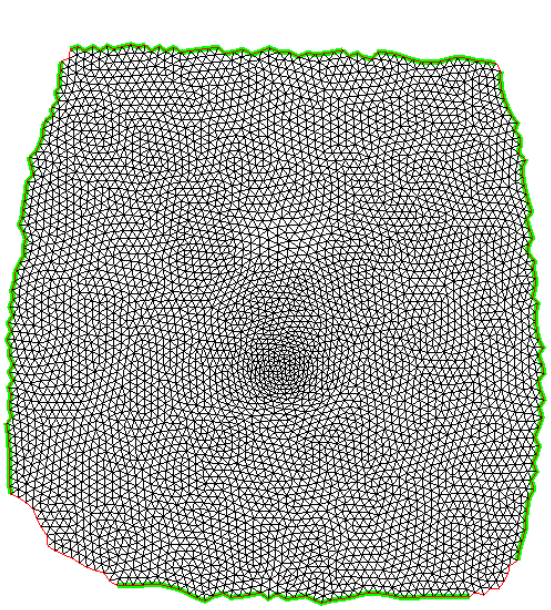


(c) *Constrained ARAP parameterization*

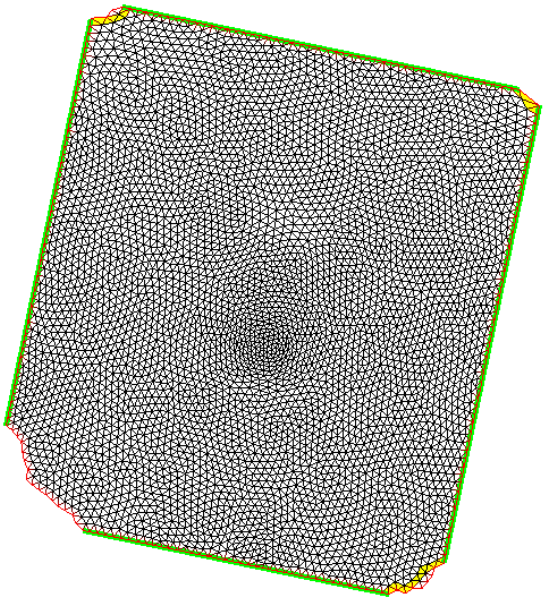
Figure 6.6: *Results for Giraffe model.*



(a) *Model with constraints*

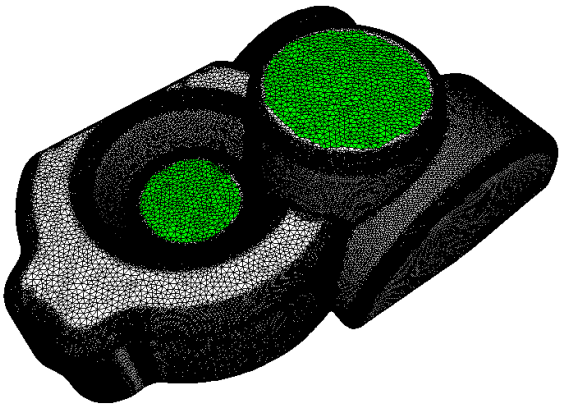


(b) *Unconstrained ARAP parameterization*

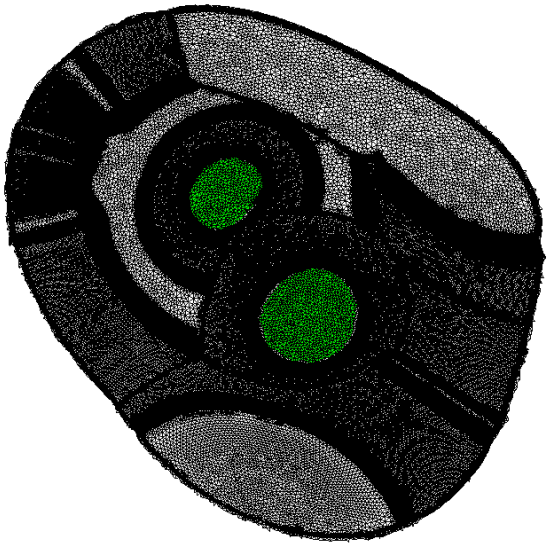


(c) *Constrained ARAP parameterization*

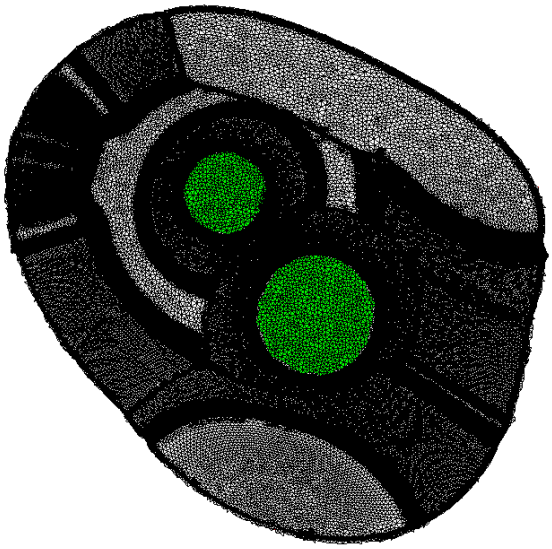
Figure 6.7: *Results for Maxface model.*



(a) *Model with constraints*



(b) *Unconstrained ARAP parameterization*



(c) *Constrained ARAP parameterization*

Figure 6.8: *Results for darmstadt model.*

Chapter 7

Enforcing constraints by iterative deformation of a parameterization

In the previous chapters we have incorporated geometric constraints into popular parameterization algorithms. In practice however, a parameterization might be already available, or the user would like to discover the different possibilities and combinations in an interactive way. This motivates our method for the iteratively modification of a planar triangulation according to given geometric constraints.

The basic outline of our approach is the following: we first take the constrained vertices and gradually modify their positions to satisfy the prescribed geometric criterion. Then, we deform the rest of the mesh accordingly.

7.1 Enforcing geometric constraints

Assume that the user has selected a set of edges or triangles and wants to enforce one of the constraints we have discussed in. If the desired shape is already present on the original mesh (i.e. we want to preserve a curve or a developable region), we simply fit that shape to the vertices in the parameterization, by solving the Orthogonal Procrustes Problem then, if we want to make the deformation more gradual, we can linearly interpolate between the old and the new positions for the vertices for intermediate steps. However, if the shape that we would like to achieve is not necessarily present, e.g when we want to deform a set of edges to a straight line, or a circle; then we choose a more refined approach involving a curvature flow.

A curvature flow is a dynamical process that gradually decreases the curvature of a curve (or surface). For an open curve this eventually results in a straight line, while a closed curve converges to a circle (assuming it makes only one net turn around any point within its interior). More formally, we would like to minimize the so-called *Willmore energy*:

$$E_W = \int \kappa^2 ds$$

where κ is the curvature of the curve.

Similarly to how Gaussian curvature is discretized on meshes; for discrete curves, the curvature is the 'angle defect' at a vertex divided the Area of the corresponding dual element (in this case the dual of a vertex is the union of the half of the two neighbouring edges, and the dual of a vertex is its midpoint):

$$\kappa_i = \frac{\varphi_i}{\frac{l_i + l_{i+1}}{2}}$$

The standard way to construct a discrete curvature flow on curves is to express κ as a function of vertex positions and apply gradient descent to E_W, an analogue of the mean curvature flow for surfaces. This method however is quite complex and very unstable: it is often the case that one must progress in extremely small steps to keep the curve from developing 'kinks'. Also, the length of the edges could change by a relatively large amount, due to numerical errors.

We instead adopt a recently published method [Crane et al., 2013a] (extending the earlier work in [Crane et al., 2011]), which allows us to take almost arbitrary large steps while remaining stable and ensures the preservation of edge lengths by construction, i.e it is an *isometric curve flow*. The reader shall refer to [Crane et al., 2013a] for details. We note that this algorithm can be generalized to surfaces to construct an angle-preserving, highly stable Willmore flow.

As this method works directly with curvature, it has to reconstruct the curve in some arbitrary reference position in the plane. To align it with the original points in the parameterization, we solve the Orthogonal Procrustes Problem [Gower and Dijksterhuis, 2004]. Assume we have vertex positions $p_0, \ldots, p_{L-1}$ and $p_0^{'}, \ldots, p_{L-1}^{'}$ for the old and new positions respectively. We first remove their mean, which, as our curve can be concave, is not computed simply as the average of the vertices but by as follows:

Then, we compute the covariance matrix C:

$$C = \sum_{i=0}^{i-1} (p' - m')(p - m)^{\mathrm{T}}.$$

Using the singular value decomposition $C = U\Sigma V^{\mathrm{T}}$, we can compute the optimal rotation aligning the two sets of points as:

$$R = UV^{\mathrm{T}}$$

We also use this procedure for aligning other features.

7.2 Deforming the parameterization

Given the new positions for some set of vertices, we want to deform the two-dimensional mesh that is our parameterization accordingly. Mesh deformation is a vast topic, we only survey a few, relatively straightforward methods here. A common approach is linear blending, which includes the well-known bone and cage-based deformation methods available in most common modeling and animation software. The basic idea is to compute a 'weighting field' around each deformable object and propagate the rigid transformations applied to them, using these weights. This approach is extremely efficient, but automatically computing the weights is a difficult task and is the topic of ongoing research, especially in the case of point-based handles.

Instead, we use a slight modification of the ARAP method presented in Chapter 6. Note that this is the context in which this algorithm was first proposed [Alexa et al., 2000], [Igarashi et al., 2005].

First, we initialize the rotations (to identity) and the edge vectors, based on the parameterization. Then, we alternate between the global and local phases, just like in Chapter 6. The only difference is that we treat the constrained vertices as constants and remove them from the linear system of the global phase. If there are previous constraints present in the parameterization, we can ensure that they remain satisfied in exactly the same way as in Chapter 6.

7.3 Results

As we build upon the ARAP algorithm, the results are practically identical to those with line or circle constraints in Chapter 6. Thus, we omit figures to conform to spatial limitations.

Chapter 8

Implementation and Performance

8.1 User Interface and data structures

For the C++ implementation of our algorithms, we have modified the Qt-based geometric framework developed by Péter Salvi for educational purposes. This framework uses the OpenMesh library [Botsch et al., 2002] for the handling of triangle meshes. OpenMesh implements the halfedge data structure, which assigns to each edge a pair of oriented halfedges, with opposite direction. This data structure allows us to easily iterate over all vertices, edges or faces connected to a vertex, etc. which is very convenient for geometry processing algorithms. On the other hand, it introduces a noticeable amount of overhead, so if performance is critical, one might use a more efficient array-based approach.

The features of OpenMesh have allowed for relatively straightforward implementation of our algorithms. The only exception is LinABF (Chapter 5), where we need to maintain a database of angles in the mesh, which we have done with the use of the hash table implementation of the STL (unordered$_m ap$).

Our user interface allows the user to add and manage any number of geometric constraints, before and after the parameterization. We also let the user move a set of handle vertices, updating the parameterization in real-time using the procedure in Chapter 6. This can be used to manually resolve global overlaps (which are much harder to explicitly prevent than local ones) in the parameterization.

8.2 Numerics

All of the presented algorithms depend on the solution of systems of linear equations that are very large (typical sizes are around several hundred thousand), but are also extremely sparse (e.g. a Laplacian matrix has 7-8 nonzeros per row on average). A long-held belief in the scientific community is that the solution of such systems requires iterative methods. This might remain true for extremely large problems (size of 10 million and above), but

for data sizes common in geometry processing it has been known for some time that sparse direct methods [Davis, 2006], such as sparse Cholesky or LU factorization tend to be considerably more efficient than iterative methods [Botsch et al., 2005]. Besides the fact that they tend to outperform Preconditioned Conjugate Gradient and its variants by a great margin (as they scale linearly), sparse direct methods also have the advantage that if we have to solve the same system with multiple right hand sides, or even with a coefficient matrix with the same non-zero pattern, we can reuse the results of the factorization and solve the system through simple forward and back-substitutions, with very small additional cost. The main disadvantage of sparse direct methods is that they have a much higher memory consumption compared to iterative methods (recall that the factor of a sparse matrix can be arbitrarily dense) and that they are much more difficult to implement. The first problem can be handled by reordering the rows and columns of the matrix which can greatly reduce the size of the factors. Finding the optimal permutation is an NP-hard problem, but one of the main reasons these methods are so successful in geometry processing is that for common matrices such as the cotangent Laplacian there are very efficient reordering heuristics available, such as Nested Dissection [George, 1973]. Compare this to iterative methods, where the analogous problem of preconditioning is much more complicated for such matrices [Krishnan et al., 2013].

For symmetric, positive-definite matrices such as the cotangent Laplacian, or the one we have to invert in ABF or ARAP initialization, we use Cholesky factorization as implemented in the open-source CHOLMOD library [Chen et al., 2008].

In LSCM and the global phase of ARAP we must solve so-called KKT systems with the structure

$$\begin{bmatrix} A & C^{\mathrm{T}} \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}.$$

Such matrices are symmetric, but indefinite, thus Cholesky solvers are not applicable[1]. As we had problems with compiling and linking sparse LU or QR factorization libraries under a Windows environment, we have instead chosen to 'export' the 'backslash' operator of MATLAB to a C++ library, which automatically selects an appropriate sparse direct solver for the system. This is a fairly efficient solution, but there are cases when it is clearly suboptimal. For example when we employ the weighting procedure in Chapter 6 to avoid overlaps, the non-zero pattern of the matrix remains the same, but as we do not use a Cholesky solver, we cannot reuse the symbolic factorization. There are many ways to solve such systems more efficiently [Benzi et al., 2005] and we have experimented with the following: notice, that if we use block Gaussian elimination, we get a block upper triangular system

$$\begin{bmatrix} A & C^{\mathrm{T}} \\ 0 & CA^{-1}C^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} b \\ d - CA^{-1}b \end{bmatrix}.$$

[1] CHOLMOD supports symmetric indefinite matrices, but not those with a KKT-structure.

If the Cholesky factorization of A is given, the computation of $CA^{-1}C^T$ requires only forward and back-substitutions[2]. As A is some kind of cotangent Laplacian, and $CA^{-1}C^T$ is symmetric positive-definite - and also much smaller than A - we can reduce the solution of KKT systems to two Cholesky factorizations, allowing us to reuse the symbolic factorization of both matrices. This method has a drawback however: $A^{-1}C^T$ can be too dense to fit in the memory. In this case, we switch back to the MATLAB-based solver.

The 2-by-2 SVD factorizations are computed using the method presented here.

8.3 Performance

The main bottleneck of all of our algorithms is the solution of sparse linear systems; so, as we employ highly efficient direct solvers for this task, we expect our algorithms to scale linearly in the size of the problem. The modifications we have made to DNCP and LinABF are relatively minor and detailed performance analysis of these algorithms can be found in the literature. Our ARAP-based method is more novel, thus we have made runtime measurements to evaluate the efficiency of our implementation. As we have observed in Chapter 6, after the initialization phase and the first global step, additional local and global iterations can be considered optional in most cases; thus, we have measured only the time of a single initialization and global phase[3]. The results, seen in Figure 8.1, confirm our earlier claims about linear scaling behavior.

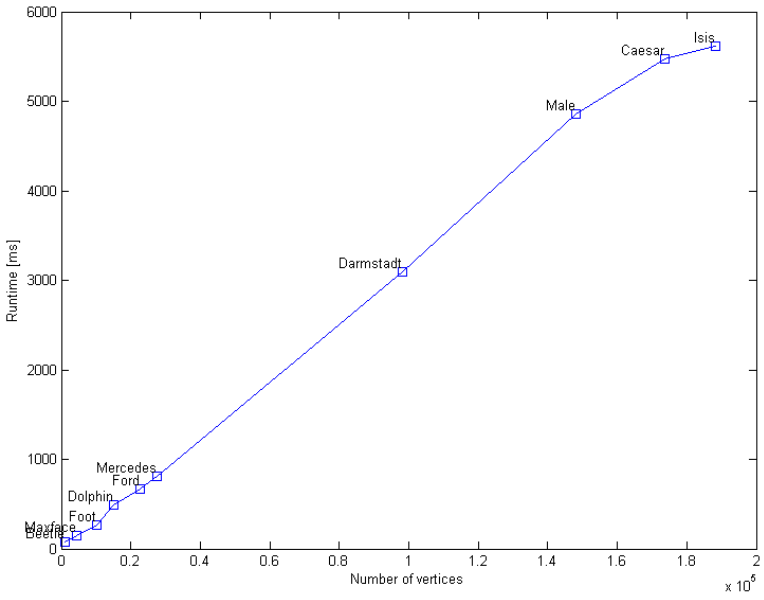


Figure 8.1: *Plot of runtime measurements for our ARAP algorithm.*

[2]CHOLMOD supports sparse matrices on the right-hand side.

[3]Note that the local phase is trivially parallelizable, meaning that by utilizing a GPU its cost can be made negligible, if necessary.

8.4 A reverse engineering application: fitting trimmed surfaces

For last, we present a practical application of our algorithms in the field of reverse engineering.

In reverse engineering our task is to reconstruct the precise CAD-model of a physical object based solely on measurements. In the typical workflow, the object is first 'scanned' into a point cloud, which is then used to generate a triangle mesh. This mesh is then segmented into different parts, to which we can fit certain geometric primitives used in CAD systems. Many man-made objects are composed from relatively simple primitives such as planes, cylinders, etc., but an increasing number of engineered geometries contain free-form surfaces, i.e. Beziér, B-Spline or NURBS patches. Besides the fact that these parametric surfaces have many unknown degrees of freedom, the patches that are present on most objects are *trimmed*, i.e. they have been cut out from a larger, unknown 4-sided parametric surface, which makes their reconstruction particularly challenging. As these are parametric surfaces, a crucial part of an fitting algorithm is finding a suitable parameterization for the patch. We demonstrate that this task can be solved with relative ease, using the framework we have developed.

We assume that the user (or some kind algorithm) has labeled parts of the boundary that are assumed to belong to the four sides of the untrimmed surface. These parts of the boundary are to be mapped to the sides of a rectangle, while the other parts are allowed to move freely within this boundary. This can be achieved by prescribing the following set of constraints:

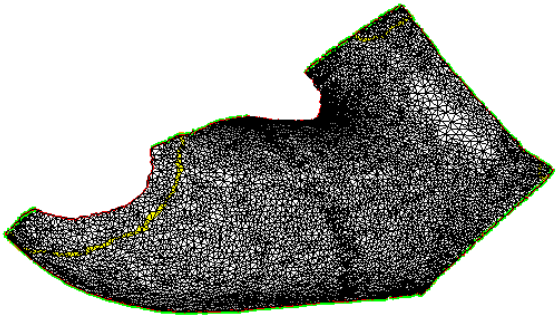
- The labeled sets of edges of the boundary shall map to straight lines.
- Edges labeled be on the same side of the rectangle shall be parallel and mapped to the same constant u or v coordinate.
- Edges belonging to adjacent sides of the rectangle shall be perpendicular.

Using our ARAP-based method, these constraints will not give the desired result however. In many cases, the regions we consider have vastly different lengths on the mesh, which means it is impossible to map them *isometrically* to a rectangle, as we attempt to do. Even if we set the lines to be similar instead of isometric, the way we enforce the orthogonality constraints in the global phase, will force them to keep their original length. However we have explicit control over the relative and absolute length of the regions (by applying scaling factors to the rotations in the global phase), which allows us to do the following:

- If two opposite sides of the rectangle are made up from a single region, we simply rescale them to have the same length (the mean of their length on the mesh, for example).

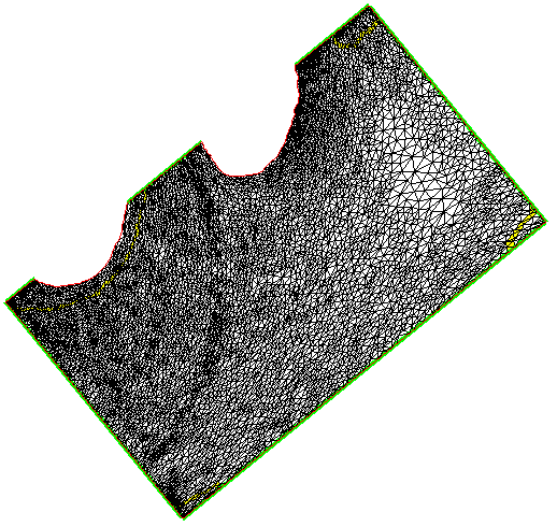
- If two opposite sides are made up from several disconnected regions, we make an estimate of their length using some heuristic, e.g. by taking the Euclidean distance of the endpoints of adjacent regions along with a conservative safety factor.

An example with a surface generated with Geomagic Studio from actual measurements is shown in Figure 8.2.



(a) *Model with constraints*

(b) *Unconstrained ARAP parameterization*



(c) *Constrained ARAP parameterization*

Figure 8.2: *Results for 4sided model.*

Conclusions and Future Work

The enforcement of high-level geometric constraints is a novel research topic in the context of mesh parameterization. After a survey of the relevant literature, we have reduced the considered constraints to low-level relations involving positions and angles, and identified three parameterization methods capable of incorporating such constraints: DNCP, ABF, and ARAP. We have extended each of these algorithms to handle geometric constraints, and evaluated their performance.

Based on our results, we have concluded that DNCP and ABF both have serious shortcomings regarding the enforcement of geometric constraints: the former is unable to handle constraints on the mesh boundary, while the latter can only constrain angles, and thus leads to incorrect results. Our ARAP-based algorithm, however, has been able to reliably handle all of the constraints we have considered, and has been found to scale linearly, as expected. Combining ARAP with a recently proposed algorithm for curvature flow, we have also developed a method to iteratively modify an existing parameterization according to geometric constraints. Finally, we have successfully applied our approach to the practical problem of parametric surface fitting.

There are many ways along which we plan to pursue this topic further in the future:

- Is it possible to make DNCP handle constraints on the boundary by using different boundary conditions?
- Can we add geometric constraints to the eigenvector-based method of [Mullen et al., 2008] in a practical way?
- We only consider constraints that lead to linear equations in the optimization variables. Can we enforce nonlinear constraints in an efficient way, as it is increasingly common in mesh deformation research [Bouaziz et al., 2012], [Deng et al., 2013]?
- We prevent overlaps and similar parameterization artifacts by a brute-force heuristic. Is there a more elegant and/or efficient solution? Can we embed our method into the optimization framework of [Schüller et al., 2013]?
- The ARAP-based iterative deformation we employ is relatively expensive, especially if we want to preserve previously existing constraints. We plan on investigating alternative methods for the solution of KKT systems and linear blending deformation, to achieve more interactive frame rates.

Bibliography

[cga,] CGAL, computational geometry algorithms library. 18

[Alexa et al., 2000] Alexa, M., Cohen-Or, D., and Levin, D. (2000). As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164. ACM Press/Addison-Wesley Publishing Co. 11, 29, 44

[Alexa and Wardetzky, 2011] Alexa, M. and Wardetzky, M. (2011). Discrete laplacians on general polygonal meshes. In *ACM Transactions on Graphics (TOG)*, volume 30, page 102. ACM. 10

[Azariadis and Aspragathos, 2001] 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. 12

[Belkin et al., 2008] Belkin, M., Sun, J., and Wang, Y. (2008). Discrete laplace operator on meshed surfaces. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 278–287. ACM. 9

[Ben-Chen et al., 2008] Ben-Chen, M., Gotsman, C., and Bunin, G. (2008). Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum*, volume 27, pages 449–458. Wiley Online Library. 11, 16

[Benko et al., 2002] 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. 13

[Bennis et al., 1991] Bennis, C., Vézien, J.-M., and Iglésias, G. (1991). Piecewise surface flattening for non-distorted texture mapping. In *ACM SIGGRAPH computer graphics*, volume 25, pages 237–246. ACM. 12

[Benzi et al., 2005] Benzi, M., Golub, G. H., and Liesen, J. (2005). Numerical solution of saddle point problems. *Acta numerica*, 14:1–137. 46

[Bettig and Hoffmann, 2011] Bettig, B. and Hoffmann, C. M. (2011). Geometric constraint solving in parametric cad. 13

[Biot, 1938] Biot, M. (1938). Theory of elasticity with large displacements and rotations. *Proc. Fifth Inr. Cong. Appl. Mech.* 30

[Bommes et al., 2013] Bommes, D., Campen, M., Ebke, H.-C., Alliez, P., Kobbelt, L., et al. (2013). Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.*, 32(4). 12

[Bommes et al., 2012] Bommes, D., Lévy, B., Pietroni, N., Puppo, E., a, C. S., Tarini, M., and Zorin, D. (2012). State of the art in quad meshing. In *Eurographics STARS*. 3

[Bommes et al., 2009] Bommes, D., Zimmer, H., and Kobbelt, L. (2009). Mixed-integer quadrangulation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 77. ACM. 4, 12, 33, 36

[Botsch et al., 2005] Botsch, M., Bommes, D., and Kobbelt, L. (2005). Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI*, pages 62–83. Springer. 46

[Botsch et al., 2010] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., et al. (2010). *Polygon mesh processing*. A K Peters/CRC Press. 10

[Botsch et al., 2002] Botsch, M., Steinberg, S., Bischoff, S., and Kobbelt, L. (2002). Openmesh-a generic and efficient polygon mesh data structure. 45

[Bouaziz et al., 2012] Bouaziz, S., Deuss, M., Schwartzburg, Y., Weise, T., and Pauly, M. (2012). Shape-up: Shaping discrete geometry with projections. In *Computer Graphics Forum*, volume 31, pages 1657–1667. Wiley. 10, 13, 29, 51

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. `http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`. 20, 25

[Brüderlin and Roller, 1998] Brüderlin, B. D. and Roller, D. (1998). *Geometric constraint solving and applications*. Springer. 13

[Chao et al., 2010] Chao, I., Pinkall, U., Sanan, P., and Schröder, P. (2010). A simple geometric model for elastic deformations. *ACM Transactions on Graphics (TOG)*, 29(4):38. 29, 30

[Chen et al., 2008] Chen, Y., Davis, T., Hager, W., and Rajamanickam, S. (2008). Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22. 46

[Cohen-Steiner and Desbrun, 2002] Cohen-Steiner, D. and Desbrun, M. (2002). Hindsight: Lscm and dcp are one and the same. *unpublished note found at: www. geometry. caltech. edu/pubs/CD02. pdf*. 20

[Courant, 1950] Courant, R. (1950). *Dirichlet's principle, conformal mapping, and minimal surfaces*. Pure and applied mathematics. Interscience Publishers. 8

[Crane et al., 2010] Crane, K., Desbrun, M., and Schröder, P. (2010). Trivial connections on discrete surfaces. In *Computer Graphics Forum*, volume 29, pages 1525–1533. Wiley Online Library. 32, 33

[Crane et al., 2011] Crane, K., Pinkall, U., and Schröder, P. (2011). Spin transformations of discrete surfaces. In *ACM Transactions on Graphics (TOG)*, volume 30, page 104. ACM. 11, 17, 43

[Crane et al., 2013a] Crane, K., Pinkall, U., and Schröder, P. (2013a). Robust fairing via conformal curvature flow. In *ACM Transactions on Graphics (TOG)*. ACM. 43

[Crane et al., 2013b] Crane, K., Weischedel, C., and Wardetzky, M. (2013b). Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph.* 20

[Davis, 2006] Davis, T. (2006). *Direct methods for sparse linear systems*, volume 2. Society for Industrial Mathematics. 46

[de Goes et al., 2013] de Goes, F., Crane, K., Desbrun, M., Schröder, P., et al. (2013). Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*, page 7. ACM. 9, 32

[Degener et al., 2003] Degener, P., Meseth, J., and Klein, R. (2003). An adaptable surface parameterization method. *IMR*, 3:201–213. 11

[Deng et al., 2013] Deng, B., Bouaziz, S., Deuss, M., Zhang, J., Schwartzburg, Y., and Pauly, M. (2013). Exploring local modifications for constrained meshes. In *Computer Graphics Forum*, volume 32, pages 11–20. Wiley Online Library. 13, 51

[Desbrun et al., 2003] Desbrun, M., Meyer, M., and Alliez, P. (2003). Intrinsic parameterizations of surface meshes. In *Computer Graphics Forum*, volume 21, pages 209–218. Wiley Online Library. 10, 16, 17

[Dong and Garland, 2007] Dong, S. and Garland, M. (2007). Iterative methods for improving mesh parameterizations. In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on*, pages 199–210. IEEE. 11

[Eck et al., 1995] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. (1995). Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182. ACM. 10

[Eckstein et al., 2001] Eckstein, I., Surazhsky, V., and Gotsman, C. (2001). Texture mapping with hard constraints. In *Computer Graphics Forum*, volume 20, pages 95–104. Wiley Online Library. 4, 12

[Fisher et al., 2007] Fisher, M., Springborn, B., Schröder, P., and Bobenko, A. (2007). An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. *Computing*, 81(2):199–213. 12

[Floater, 1997] Floater, M. (1997). Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250. 10

[Floater, 2003] Floater, M. (2003). Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27. 9, 10

[Floater and Hormann, 2005] Floater, M. and Hormann, K. (2005). Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling*, pages 157–186. 10

[Gal et al., 2009] Gal, R., Sorkine, O., Mitra, N. J., and Cohen-Or, D. (2009). iwires: an analyze-and-edit approach to shape manipulation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 33. ACM. 13

[Gander et al., 1989] Gander, W., Golub, G. H., and von Matt, U. (1989). A constrained eigenvalue problem. *Linear Algebra and its applications*, 114:815–839. 17

[George, 1973] George, A. (1973). Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363. 46

[Golub et al., 2000] Golub, G. H., Zhang, Z., and Zha, H. (2000). Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the lanczos process with inner–outer iterations. *Linear Algebra and its Applications*, 309(1):289–306. 17

[Gopin, 1978] Gopin, A. (1978). *Development of a dimension-based data structure for twodimensional computer graphics*. PhD thesis, Masters thesis, MIT. 13

[Gower and Dijksterhuis, 2004] Gower, J. C. and Dijksterhuis, G. B. (2004). *Procrustes problems*, volume 3. Oxford University Press. 29, 43

[Grinspun et al., 2008] Grinspun, E., Desbrun, M., Polthier, K., Schröder, P., and Stern, A. (2008). Discrete differential geometry: an applied introduction. *ACM SIGGRAPH Course*. 11

[Guo et al., 2000] Guo, Y., Batoz, J., Naceur, H., Bouabdallah, S., Mercier, F., and Barlet, O. (2000). Recent developments on the analysis and optimum design of sheet metal forming parts using a simplified inverse approach. *Computers & Structures*, 78(1):133–148. 12

[Habbecke and Kobbelt, 2012] Habbecke, M. and Kobbelt, L. (2012). Linear analysis of nonlinear constraints for interactive geometric modeling. In *Computer Graphics Forum*, volume 31, pages 641–650. Wiley Online Library. 13

[Hoffmann and Joan-Arinyo, 2005] Hoffmann, C. M. and Joan-Arinyo, R. (2005). A brief on constraint solving. *Computer-Aided Design and Applications*, 2(5):655–663. 13

[Hormann and Greiner, 2000] Hormann, K. and Greiner, G. (2000). Mips: An efficient global parametrization method. Technical report, DTIC Document. 11, 17

[Hormann et al., 2007] Hormann, K., Lévy, B., Sheffer, A., et al. (2007). Mesh parameterization: Theory and practice. *SIGGRAPH Course Notes*. 6, 10, 17

[Igarashi et al., 2005] Igarashi, T., Moscovich, T., and Hughes, J. F. (2005). As-rigid-as-possible shape manipulation. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1134–1141. ACM. 11, 29, 44

[Igarashi et al., 2009] Igarashi, Y., Igarashi, T., and Suzuki, H. (2009). Interactive cover design considering physical constraints. In *Computer Graphics Forum*, volume 28, pages 1965–1973. Wiley. 12, 29

[Jermann et al., 2006] Jermann, C., Trombettoni, G., Neveu, B., and Mathis, P. (2006). Decomposition of geometric constraint systems: a survey. *International Journal of Computational Geometry & Applications*, 16(05n06):379–414. 13

[Kälberer et al., 2007] Kälberer, F., Nieser, M., and Polthier, K. (2007). Quadcover-surface parameterization using branched coverings. In *Computer Graphics Forum*, volume 26, pages 375–384. Wiley Online Library. 33

[Kami et al., 2005] Kami, Z., Gotsman, C., and Gortler, S. J. (2005). Free-boundary linear parameterization of 3d meshes in the presence of constraints. In *Shape Modeling and Applications, 2005 International Conference*, pages 266–275. IEEE. 10, 12

[Kharevych et al., 2006] Kharevych, L., Springborn, B., and Schröder, P. (2006). Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)*, 25(2):412–438. 11

[Kós and Várady, 2003] Kós, G. and Várady, T. (2003). Parameterizing complex triangular meshes. *Curve and surface design*, pages 265–274. 10

[Kraevoy et al., 2003] Kraevoy, V., Sheffer, A., and Gotsman, C. (2003). Matchmaker: constructing constrained texture maps. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 326–333. ACM. 4, 12

[Krishnan et al., 2013] Krishnan, D., Fattal, R., and Szeliski, R. (2013). Efficient preconditioning of laplacian matrices for computer graphics. volume 28, pages 1–10, New York, NY, USA. ACM. 46

[Lee et al., 2008] Lee, T.-Y., Yen, S.-W., and Yeh, I.-C. (2008). Texture mapping with hard constraints using warping scheme. *Visualization and Computer Graphics, IEEE Transactions on*, 14(2):382–395. 12

[Lee et al., 2002] Lee, Y., Kim, H. S., and Lee, S. (2002). Mesh parameterization with a virtual boundary. *Computers & Graphics*, 26(5):677–686. 10

[Lévy et al., 2002] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 362–371. ACM. 10, 16, 17, 19, 20

[Li et al., 2010] Li, B., Zhang, X., Zhou, P., and Hu, P. (2010). Mesh parameterization based on one-step inverse forming. *Computer-Aided Design*, 42(7):633–640. 12

[liang Chen et al., 2011] liang Chen, W., Wei, P., and Bao, Y. (2011). Surface Flattening based on Linear-Elastic Finite Element Method. 5(7):728 – 734. 12

[Light and Gossard, 1982] Light, R. and Gossard, D. (1982). Modification of geometric models through variational geometry. *Computer-Aided Design*, 14(4):209–214. 13

[Lipman, 2012] Lipman, Y. (2012). Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics (TOG)*, 31(4):108. 12

[Lipman, 2013] Lipman, Y. (2013). Construction of Injective Mappings Of Meshes. *arXiv preprint arXiv:1310.0955*. 12

[Liu et al., 2008] Liu, L., Zhang, L., Xu, Y., Gotsman, C., and Gortler, S. (2008). A local/-global approach to mesh parameterization. In *Computer Graphics Forum*, volume 27, pages 1495–1504. Wiley Online Library. 11, 17, 29, 30, 31

[Maillot et al., 1993] Maillot, J., Yahia, H., and Verroust, A. (1993). Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 27–34. ACM. 11, 17

[Masuda et al., 2007] Masuda, H., Yoshioka, Y., and Furukawa, Y. (2007). Preserving form features in interactive mesh deformation. *Computer-Aided Design*, 39(5):361–368. 13

[Mitra et al., 2012] Mitra, N. J., Wand, M., Zhang, H., Cohen-Or, D., and Bokeloh, M. (2012). Structure-aware shape processing. In *Eurographics 2013-State of the Art Reports*, pages 175–197. The Eurographics Association. 13

[Mullen et al., 2011] Mullen, P., Memari, P., de Goes, F., and Desbrun, M. (2011). Hot: Hodge-optimized triangulations. *ACM Transactions on Graphics (TOG)*, 30(4):103. 12

[Mullen et al., 2008] Mullen, P., Tong, Y., Alliez, P., and Desbrun, M. (2008). Spectral conformal parameterization. In *Computer Graphics Forum*, volume 27, pages 1487–1494. Wiley Online Library. 10, 17, 20, 51

[Myles and Zorin, 2012] Myles, A. and Zorin, D. (2012). Global parametrization by incremental flattening. *ACM Transactions on Graphics (TOG)*, 31(4):109. 11, 16, 31

[Myles and Zorin, 2013] Myles, A. and Zorin, D. (2013). Controlled-distortion constrained global parametrization. *ACM Trans. Graph.*, 32(4):xx–xx. 4

[Needham, 1999] Needham, T. (1999). *Visual complex analysis*. Oxford University Press, USA. 8, 17

[Parikh and Boyd, 2013] Parikh, N. and Boyd, S. (2013). Proximal algorithms. 29

[Pinelis, 2005] Pinelis, I. (2005). Cyclic polygons with given edge lengths: existence and uniqueness. *Journal of Geometry*, 82(1-2):156–171. 15

[Pinkall and Polthier, 1993] Pinkall, U. and Polthier, K. (1993). Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36. 6, 9, 10

[Ray et al., 2008] Ray, N., Vallet, B., Li, W. C., and Lévy, B. (2008). N-symmetry direction field design. *ACM Transactions on Graphics (TOG)*, 27(2):10. 33

[Sander et al., 2002] Sander, P. V., Gortler, S. J., Snyder, J., and Hoppe, H. (2002). Signal-specialized parametrization. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 87–98. Eurographics Association. 11

[Sander et al., 2001] Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. (2001). Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM. 11, 17

[Schüller et al., 2013] Schüller, C., Kavan, L., Panozzo, D., and Sorkine-Hornung, O. (2013). Locally injective mappings. In *Computer Graphics Forum*, volume 32, pages 125–135. Wiley Online Library. 12, 51

[Sheffer and de Sturler, 2001] Sheffer, A. and de Sturler, E. (2001). Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers*, 17(3):326–337. 11, 16, 23

[Sheffer et al., 2005] Sheffer, A., Lévy, B., Mogilnitsky, M., and Bogomyakov, A. (2005). Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG)*, 24(2):311–330. 11, 24, 25

[Sheffer et al., 2006] Sheffer, A., Praun, E., and Rose, K. (2006). Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):105–171. 10

[Sorkine and Alexa, 2007] Sorkine, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *ACM International Conference Proceeding Series*, volume 257, pages 109–116. Citeseer. 11, 13, 29

[Sorkine et al., 2002] Sorkine, O., Cohen-Or, D., Goldenthal, R., and Lischinski, D. (2002). Bounded-distortion piecewise mesh parameterization. In *Proceedings of the conference on Visualization'02*, pages 355–362. IEEE Computer Society. 11

[Springborn et al., 2008] Springborn, B., Schröder, P., and Pinkall, U. (2008). Conformal equivalence of triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 27, page 77. ACM. 11, 16, 17

[Sutherland, 1964] Sutherland, I. E. (1964). Sketchpad: a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, pages 6–329. ACM. 13

[Tang et al., 2003] Tang, Y., Wang, J., Bao, H., and Peng, Q. (2003). Rbf-based constrained texture mapping. *Computers & Graphics*, 27(3):415–422. 12

[Tutte, 1963] Tutte, W. (1963). How to draw a graph. *Proc. London Math. Soc*, 13(3):743–768. 10

[Vallet and Lévy, 2009] Vallet, B. and Lévy, B. (2009). What you seam is what you get. Technical report, INRIA - ALICE Project Team. 12, 16

[Varady et al., 1997] Varady, T., Martin, R. R., and Cox, J. (1997). Reverse engineering of geometric modelsan introduction. *Computer-Aided Design*, 29(4):255–268. 3

[Wang, 2008] Wang, C. C. (2008). WireWarping: A fast surface flattening approach with length-preserved feature curves. *Computer-Aided Design*, 40(3):381–395. 12

[Wardetzky et al., 2007] Wardetzky, M., Mathur, S., Kalberer, F., and Grinspun, E. (2007). Discrete laplace operators: no free lunch. In *ACM International Conference Proceeding Series*, volume 257, pages 33–37. 9

[Yu et al., 2012] Yu, H., Lee, T.-Y., Yeh, I.-C., Yang, X., Li, W., and Zhang, J. J. (2012). An rbf-based reparameterization method for constrained texture mapping. *Visualization and Computer Graphics, IEEE Transactions on*, 18(7):1115–1124. 12

[Zayer et al., 2007] Zayer, R., Lévy, B., Seidel, H., et al. (2007). Linear angle based parameterization. In *Fifth Eurographics Symposium on Geometry Processing-SGP 2007*, pages 135–141. 11, 16, 24

[Zheng et al., 2011] Zheng, Y., Fu, H., Cohen-Or, D., Au, O. K.-C., and Tai, C.-L. (2011). Component-wise controllers for structure-preserving shape manipulation. In *Computer Graphics Forum*, volume 30, pages 563–572. Wiley Online Library. 13