Low Distortion Surface Parameterization

Dissertationsschrift

vorgelegt von
Dipl.-Math. Felix Kälberer

am Fachbereich
Mathematik und Informatik
der Freien Universität Berlin

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Gutachter: Prof. Dr. Konrad Polthier
Prof. Dr. Craig Gotsman

Einreichung: 29. August 2013
Disputation: 17. Dezember 2013

Acknowledgments

I am grateful to my advisor Konrad Polthier for his continuous support and encouragement over many years, and for providing an excellent research environment in Berlin. I thank my friend and colleague Matthias Nieser for thousands of joyful and productive hours we have worked together, and for his ideas and endless energy in our common research and recreational activities.

I thank Ulrich Reitebuch for his continuous help and I thank Christoph von Tycowicz for driving forward our joint publications. My thanks also go to Klaus Hildebrandt, Carsten Lange and Isabella Thiesen, for helpful discussions and their support for this work, and all other dear colleagues who made me enjoy my research. I am still thankful to Max Wardetzky for his insights and for being a role model.

I deeply thank my soon-to-be wife Kerstin for her infinite love and patience in the recent years. My thanks are similarly due to my mother and my family for their continuous support.

This work was supported by mental images GmbH (now NVIDIA ARC GmbH) and the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin.

Contents

Verfassererklärung

Gemäß §7 (4) der Promotionsordnung versichere ich hiermit, diese Arbeit selbständig verfasst zu haben. Ich habe alle bei der Erstellung dieser Arbeit benutzten Hilfsmittel und Hilfen angegeben.

Felix Kälberer

Introduction

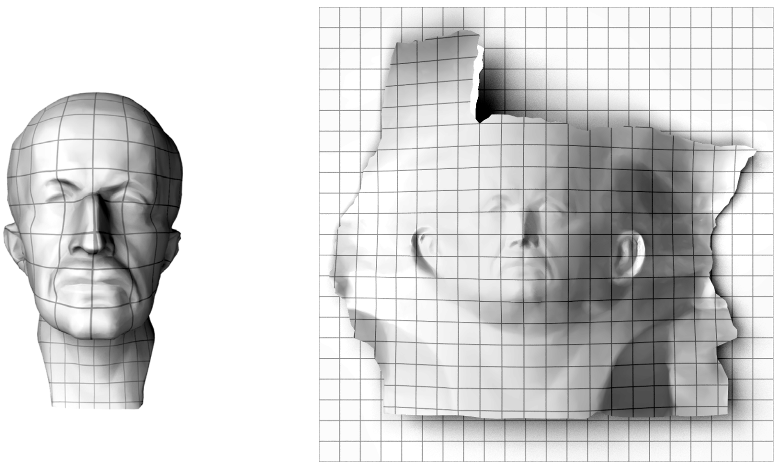


Figure 1: The unfolding of the Max Planck head model into the plane.

Surface parameterization

Surface parameterization is the process of mapping a surface into the Euclidean plane. The inverse process maps a region of the plane onto the surface, and along with it, much of the structure of the Euclidean plane is transferred to the surface. For example, surface parameterization induces a coordinate system on the surface, which has applications in texture mapping, scattered data fitting, reverse engineering, and modeling.

By considering all points with at least one integral coordinate, we can define a unit grid on the parameterized surface. The unit grid divides the surface into quadrilateral patches and so has its own applications, such as the remeshing of surfaces into regular quadrilaterals. Moreover, the unit grid also lends itself for an intuitive visualization of the parameterization map, in which the parameterization's quality can be quickly captured. The requirements on the quality differ from application to application, but most often, a low distortion—measured in angle, length and area distortion—is among the main requirements.

While angle-preserving, length-preserving and area-preserving parameterizations exist, the three properties can generally not be achieved at the same time. The exact preservation of one of these measures often leads to large distortions of the other two, and thus, a tradeoff between them is often preferred.

Next to distortion, the alignment of the parameterization is crucial for many applications. Consider texture synthesis, remeshing, or architectural designs: there are generally application dependent reasons to guide the unit grid lines in certain directions due to numerical, structural, or aesthetic reasons. The central algorithm of this work is the QuadCover algorithm, which I developed with Matthias Nieser under the supervision of Konrad Polthier. It is tailored to satisfy the directional requirements and it finds a best-fitting parameterization to given input directions.

The QuadCover algorithm, as published in 2007 [KNP07], sets up a clean mathematical concept that shows how piecewise linear direction fields with fractional indices can be regarded as vector fields on branched covering surfaces. This enables to apply the wealth of well known vector field concepts to direction fields, such as the notion of harmonicity and Hodge-Helmholtz decomposition. With this concept, QuadCover was the first automatic parameterization method to create clean quadrilateral mesh layouts from direction fields.

A wealth of applications makes high demands on parameterizations. The requirements include layout constraints, interactivity, quality and speed, and so, many improvements were developed for QuadCover and other parameterization methods. Together with Matthias Nieser, I have intensively developed the algorithm since the original publication: We have extended QuadCover to other symmetry orders [KNP10, NPPZ12], forced parameter lines exactly to surface features (Matthias Nieser [Nie12]), and developed additional applications for QuadCover in compression together with Christoph von Tycowicz [vTKP11]. Furthermore, I developed optimizations to generate extremely regular QuadCover parameterizations. These extensions are not published yet. Most notably, I have contributed the following:

Development of QuadCover With QuadCover [KNP07], we developed the first parameterization algorithm to automatically produce regular, feature-aligned quad layouts. Joint work with Matthias Nieser.

Generation of smooth direction fields A robust construction method for smooth direction fields and a framework to naturally edit the branch point positions while maintaining its smoothness. Joint work with Matthias Nieser.

Extremely regular parameterizations I have developed new methods to drastically reduce the parameterization's distortion. Significant reduction of distortion is made during the enforcement of both *local* and *global* continuity and the distortion of the two is *optimally balanced* by choosing the right number of branch points. The key ingredients are:

– *Curl minimizing branch points.* Local distortion is due to the curl of the guiding field and is mainly influenced by the positions of branch points. Their optimization is often a game of trial and error, which makes the quick evaluation of curl essential. The very fast energy evaluation via complex sparse Cholesky updates, paired with an intelligent choice of promising candidates in the search space makes brute force search finally feasible. The distortion is well below that of recent state-of-the-art methods [MZ13].

– *Thorough global rounding* A thorough and practical heuristic for the NP-hard problem of optimally connecting parameter lines. Our heuristic includes the computation of a shortest basis of homology generators on the covering in [KNP10] and a fast method to compute the covering surface's period matrix. The latter allows to speed up the search for good solutions by several orders of magnitude.

– *The optimal number of branch points.* While the minimization of curl favors many branch points, a high number of branch points increases the artifacts of global continuity enforcement. Thus, there is an optimal number of branch points, which depends on the parameter line density. By giving an empirical estimate of the expected error, the right number of branch points can be chosen for each parameterization automatically.

This thesis is structured as follows. In Chapter 1, we give an in-depth introduction the mathematical basics of discrete surfaces, vector fields calculus, and some algebraic topology. We are taking a non-standard route to piecewise linear surfaces, extending common definitions, to avoid deficiencies when dealing with self-intersecting branched covering surfaces.

The QuadCover algorithm itself is explained in Chapters 2 to 5. We put an emphasis on the completeness of the description, including many details that influence QuadCover's performance or ease of implementation.

Chapter 6 contains the extensions to optimize the position and number of branch points for low-distortion parameterizations, and Chapter 7 summarizes the results and compares them to competing methods.

Chapter 1

Foundations

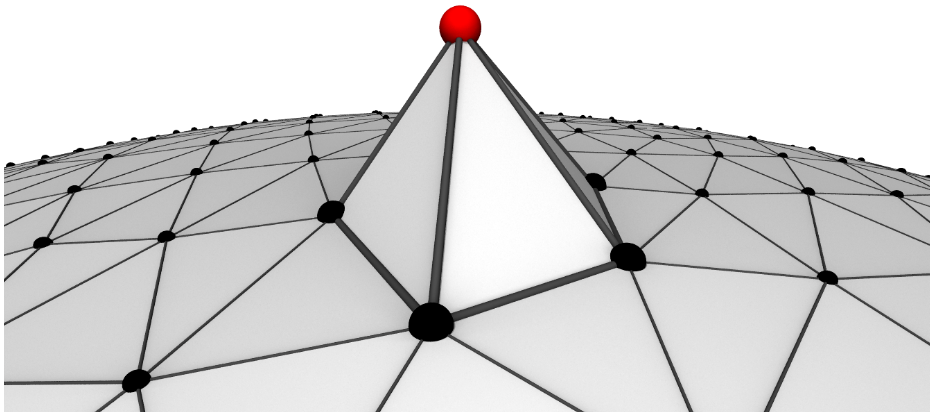


Figure 1.1: Vertices, edges, and triangles of a simplicial complex.

In this chapter we will introduce all major mathematical objects that we need for the QuadCover parameterization. This includes the parameterization domain—piecewise linear surfaces—along with the discrete objects living on these surfaces: piecewise linear functions together with the operators acting on them, as well as piecewise constant vector fields, whose calculus then bridges the gap to the Hodge-Helmholtz theorem and algebraic topology.

1.1 Piecewise Linear Surfaces

Most of the mathematical objects in this thesis live on triangle meshes—or *simplicial surfaces*. In the description of simplicial surfaces we differentiate between their abstract combinatorial structure and their geometric immersion in Euclidean space. This accords to the clearly distinguished treatment of connectivity and geometry in practical mesh processing. Compared to a purely geometric description of simplicial complexes found in many textbooks such as Ziegler [Zie98], our approach allows a larger variety of shapes including self-intersecting surfaces and degenerated triangles. This generalization is necessary, because our construction of branched covering surfaces in Section 3.2 will inevitably lead to self-intersecting surfaces, which makes any standard definition inappropriate.

1.1.1 Abstract Simplicial Complexes

We will begin with the definition of a (finite) abstract simplicial complex, which generalizes abstract points, line segments, triangles, and tetrahedra to arbitrary dimensions. Our definitions adhere closely to Polthier and Munkres [Pol02, Mun84] where possible.

Definition 1.1 *Let $\mathfrak{V}$ be a finite set of abstract points. An* ***abstract simplicial complex*** *$\mathfrak{K}$ is a set of non-empty subsets of $\mathfrak{V}$ such that if σ is an element of $\mathfrak{K}$, so is every non-empty subset of σ.*

An element $\sigma \in \mathfrak{K}$ is called a **simplex** of $\mathfrak{K}$ and its **dimension** is $|\sigma| - 1$. Each non-empty (proper) subset of σ is called a (proper) **face** of σ. The dimension of the complex $\mathfrak{K}$ is given by the largest dimension of any simplex of $\mathfrak{K}$. A simplex is called **vertex**, **edge**, or **triangle** if it has dimension zero, one, or two, respectively. The **boundary** of $\mathfrak{K}$ is formed by any simplex τ that is a proper face of exactly one simplex of $\mathfrak{K}$, together with the faces of τ. An abstract simplicial complex is **closed** if its boundary is empty.

The concept of complexes allows an immediate definition of neighborhood relations among the simplices. Two simplices of different dimension of a simplicial complex $\mathfrak{K}$ are called **incident**, if one of them is a face of the other. We will write $\sigma \in \tau$ if σ is a face of τ. One calls two distinct d-dimensional simplices σ_1 and σ_2 **adjacent**, denoted $\boldsymbol{\sigma_1 \sim \sigma_2}$, if they either share a common $d-1$-dimensional face, or if they are both face of a common $d+1$-dimensional simplex of $\mathfrak{K}$. Two abstract simplicial complexes $\mathfrak{V}$ and $\mathfrak{W}$ are said to be **isomorphic**, if there is a bijective map f taking the abstract points of $\mathfrak{V}$ to those of $\mathfrak{W}$, such that $\{\mathfrak{v}_1, \ldots, \mathfrak{v}_{m+1}\} \in \mathfrak{V}$ if and only if $\{f(\mathfrak{v}_1), \ldots, f(\mathfrak{v}_{m+1})\} \in \mathfrak{W}$.

The **star** of a simplex σ is the smallest subcomplex containing all simplices that contain σ. The **link** of σ is the subset of simplices of star σ that do not contain any faces of σ. For example, if σ is an interior vertex of $\mathfrak{K}$, the link of σ is the boundary of the star of σ. The neighborhood relations allow the definition of a topological surface, compare [Haz93].

Definition 1.2 *An* ***abstract simplicial surface*** *M is a simplicial complex consisting of a finite set of triangles (and their faces) such that*

1. *each edge is incident to either one or two triangles.*
2. *around each vertex v of M the faces incident to v form a cycle of faces sequentially adjacent to one another.*

The cycle around a vertex is closed, if and only if the vertex does not lie on the boundary. Definition 1.2 (2.) implies that every boundary vertex is incident to exactly two boundary edges: one edge is contained in the first triangle of the cycle of faces, the other one is contained in the last face. Thus, the boundary of a surface must form one or several closed loops, the **boundary components**.

It is possible to equip each triangle with an orientation by ordering its edges, such that they form a closed loop. Two adjacent triangles have the same orientation, if their common incident edge inherits different orientations from both triangles. A simplicial surface is **orientable**, if it is possible to orient its triangles such that any two adjacent triangles have the same orientation.

The **valence** or **degree** of a vertex refers to the number of its incident edges. The sets of all vertices, edges, and faces of a simplicial surface will be denoted with V, E, F, and their respective cardinalities will be denoted v, e, and f. The **Euler characteristic** of a simplicial surface M is defined by

$$\chi(M) = \mathsf{v} - \mathsf{e} + \mathsf{f}. \tag{1.1}$$

We use the Euler characteristic to define the **genus** g of a closed abstract simplicial surface through the relation

$$2 - 2\mathsf{g}(M) = \chi(M). \tag{1.2}$$

1.1.2 Simplicial Surfaces

So far, we have focused only on the connectivity of simplicial complexes, ignoring any geometric associativity. We now move our focus to geometric realizations of simplicial complexes. To account for practical applications, where self-intersecting simplicial surfaces are commonplace, we will consider simplicial complexes in a sufficiently high-dimensional Euclidean space $\mathbb{R}^N$ and later map the geometry from this embedding to $\mathbb{R}^3$. This allows us to carry over the induced topology from $\mathbb{R}^N$, in which the simplicial complex always has a self-intersection free embedding, which is later required for a proper construction of the topology.

First, let $\{v_1, \ldots, v_{m+1}\}$ be a set of geometric independent points in $\mathbb{R}^N$, *i. e.*, points for which the vectors

$$v_2 - v_1,\ v_3 - v_1,\ \ldots,\ v_{m+1} - v_1$$

are linearly independent.

Definition 1.3 *The (geometric)* ***m-simplex*** Δ^m *spanned by* $v_1, \ldots, v_{m+1}$ *is the convex hull of* $\{v_1, \ldots, v_{m+1}\}$*, formally,*

$$\Delta^m = \left\{ \sum_{i=1}^{m+1} \lambda_i v_i \;\middle|\; 0 \leq \lambda_i \leq 1,\ \sum_{i=1}^{m+1} \lambda_i = 1 \right\}.$$

For each $x \in \Delta^m$, the numbers λ_i are uniquely determined and are called the **barycentric coordinates** of the point x with respect to $v_1, \ldots, v_{m+1}$. A **face** of Δ^m is any simplex spanned by a non-empty subset of $\{v_1, \ldots, v_{m+1}\}$. Multiple simplices are structured in a simplicial complex:

Definition 1.4 *A* ***(geometric) simplicial complex*** K *is a collection of simplices in* $\mathbb{R}^N$*, such that*

1. *every face of a simplex in* K *is in* K*.*
2. *the intersection of any two simplices in* K *is a face of each of them.*

The abstract and geometric simplicial complexes are related in the following sense. Let K be a geometric simplicial complex, and V be the set of vertices of K. Let $\mathfrak{K}$ be the collection of all subsets $\{v_{i_1}, \ldots, v_{i_k}\}$ of V such that $v_{i_1}, \ldots, v_{i_k}$ span a simplex of K. Then $\mathfrak{K}$ is an abstract simplicial complex and is called the **vertex scheme** of K.

Vice versa, if the vertex scheme of a simplicial complex K is isomorphic to a given abstract simplicial complex $\mathfrak{K}$, we call K a **geometric realization** of $\mathfrak{K}$. There is always a standard realization for an abstract simplicial complex $\mathfrak{K}$:

Definition 1.5 (Standard realization) *Let* $\mathfrak{K}$ *be an abstract simplicial complex with vertices* $\mathfrak{v}_1, \ldots, \mathfrak{v}_{\mathsf{v}}$*. Then the assignment* $\mathfrak{v}_i \mapsto e_i$ *of the abstract vertices to the standard basis vectors* $\{e_1, \ldots, e_{\mathsf{v}}\}$ *of* $\mathbb{R}^{\mathsf{v}}$ *and the assignment of each abstract simplex* $\{\mathfrak{v}_{i_1}, \ldots, \mathfrak{v}_{i_k}\}$ *to a geometric simplex spanned by* $\{e_{i_1}, \ldots, e_{i_k}\}$ *yields a geometric simplicial complex whose vertex scheme is isomorphic to* $\mathfrak{K}$*. It is called the* ***standard realization*** *of* $\mathfrak{K}$*.*

The surfaces that we consider in this thesis are living in $\mathbb{R}^3$. The following definition will relate the high-dimensional standard realization to the triangle meshes as they are known from computational geometry. In this thesis, we will always assume that the dimension n of the ambient space is 3, although this restriction is not necessary.

Definition 1.6 (Triangle mesh) *Let* $\mathfrak{K}$ *be an abstract simplicial surface with* v *vertices and let* K *be the standard realization of* $\mathfrak{K}$ *(with vertices* $\{v_1 = e_1,\ \ldots,\ v_{\mathsf{v}} = e_{\mathsf{v}}\} \subset \mathbb{R}^{\mathsf{v}}$*).*

*A **triangle mesh** $M = (\mathfrak{K}, P)$ is a an abstract simplicial surface $\mathfrak{K}$ together with a set of points $P = \{p_1, \dots, p_\mathsf{v}\} \subset \mathbb{R}^n$ and the (unique) linear map $\varphi : \mathbb{R}^\mathsf{v} \to \mathbb{R}^n$ so that*

1. $p_{i_1}, \dots, p_{i_k}$ are geometrically independent if $\{\mathfrak{v}_{i_1}, \dots, \mathfrak{v}_{i_k}\} \in \mathfrak{K}$
2. $\varphi(v_i) = p_i$

The linear map φ maps the vertices of K to positions $p_i \in \mathbb{R}^n$, but also maps the interior points of simplices of K into simplices in $\mathbb{R}^n$. The uniqueness of φ (and its one-to-one correspondence with P) is obvious when written in matrix form:

$$\varphi = \begin{pmatrix} p_1 & \dots & p_\mathsf{v} \end{pmatrix},$$

where the vectors $p_1, \dots, p_\mathsf{v}$ constitute the columns of the matrix φ. Barycentric coordinates are preserved by φ, although they are not necessarily unique in $\varphi(K)$.

The terminology *triangle mesh* is used in computer graphics and computational geometry practice, where the connectivity of triangles are treated distinctively from the vertex positions. This distinction in practice allows arbitrary vertex positions and intersections of non-adjacent simplices, so this possibility should be taken into account when working with surfaces. Due to self-intersections of $\varphi(K)$ in $\mathbb{R}^3$, K might lose its property of being a simplicial complex under the application of φ. Thus, being finicky at this point, we will induce the surface's topology from the self-intersection-free embedding K in the higher-dimensional space $\mathbb{R}^\mathsf{v}$, but stick to the surface's intuitive metric induced from its point positions in $\mathbb{R}^3$ as explained below.

Let the **underlying space** $|K| \subset \mathbb{R}^\mathsf{v}$ denote the union of all simplices in K. We define a topology on $|K|$ as the subspace topology inherited from the surrounding space $\mathbb{R}^\mathsf{v}$. Following Gromov and Wardetzky [Gro99, War06] we define a distance metric g on $|K|$ via the specification of the length of curves in $|K|$, and measure distances of points of $|K|$ via g:
If $\gamma : [a, b] \to |K|$ is a continuous curve, then the **length** of γ (with respect to φ) is the supremum over all *admissible* partitions $Z = \{t_0 = a \leq t_1 \leq \dots \leq t_k = b\}$ of $[a, b]$:

$$l(\gamma) := \sup_Z \sum_{i=1}^{k} d_{\mathbb{R}^n}\big(\varphi \circ \gamma(t_{i-1}), \varphi \circ \gamma(t_i)\big).$$

A partition is admissible if $\gamma(t_{i-1})$ and $\gamma(t_i)$ lie in the same triangle T of K (possibly on the boundary of T). Here $d_{\mathbb{R}^n}$ denotes the Euclidean distance in $\mathbb{R}^n$. The distance between two points $x, y \in |K|$ is defined as

$$d(x, y) := \inf_\gamma l(\gamma),$$

the infimum taken over all continuous curves $\gamma : [a, b] \to |K|$ connecting x and y. We set $d(x, y) = \infty$ if there is no path of finite length from x to y. The distance d also induces a metric tensor $g_{|K|}$ on $|K|$, arising as the derivative of d.

This metric tensor necessarily agrees with the Euclidean metric tensor on the interior of triangles, as the triangles themselves are flat. Indeed $g_{|K|}$ also coincides with the Euclidean metric across the interior of edges, as each edge star can be flattened into the plane without changing the intrinsic length.

If $\varphi(|K|) \subset \mathbb{R}^n$ is a simplicial complex (*i. e.*, it has no self intersections), then we do not need to take the detour and define the topology and metric on $|K| \subset \mathbb{R}^{\mathsf{v}}$. Instead, we can induce the *same* topology and metric on $\varphi(|K|)$ directly from $\mathbb{R}^n$. If this is the case, or if there is no risk of confusion, we will drop the distinction between a triangle mesh as defined above, stick to the much simpler definition of a geometric simplicial surface:

Definition 1.7 *A **(geometric) simplicial surface** is a geometric simplicial complex with vertices in $\mathbb{R}^n$, whose vertex scheme is an abstract simplicial surface.*

For the rest of this thesis, we will use M to denote either a simplicial surface or a triangle mesh. The distinction between the two definitions becomes irrelevant, as both describe surfaces with the same essential structures that we need in this thesis: the discrete structure of a simplicial complex, outfitted with a topology and a metric that is flat everywhere but at the vertices.

To differentiate between abstract and geometric simplices, we will use italic letters $p, q, \ldots \in \mathsf{V}, e, f, \ldots \in \mathsf{E}$ and $s, t, \ldots \in \mathsf{F}$ to denote abstract simplices, and use bold face letters to denote their geometric realizations, for example $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$, and sometimes $\mathbf{e} = \mathbf{p} - \mathbf{q} \in \mathbb{R}^3$. When in doubt, please also check the general notes in the List of Symbols at the end of this thesis.

Various measures of curvature have been examined in differential geometry, and many curvature concepts have their discrete counter-part. For example, the Gauss curvature is defined in Polthier [Pol02] as follows:

Curvature

Definition 1.8 (Gauss curvature). *Let M be a simplicial surface and $p \in M$ a vertex. Let $\{t_1, \ldots, t_k\}$ be the set of faces of* $\operatorname{star}(p)$ *and α_i be the vertex angle of the triangle t_i at the vertex p as in Figure 1.2. Then the discrete **Gauss curvature** $K(p)$ of an interior vertex p on a simplicial surface M is defined as the vertex angle excess*

$$K(p) = 2\pi - \sum_{i=1}^{k} \alpha_i.$$

Interior points p of a face or of an open edge have a neighborhood which is isometric to a planar Euclidean domain, and we define $K(p) = 0$ in these cases.

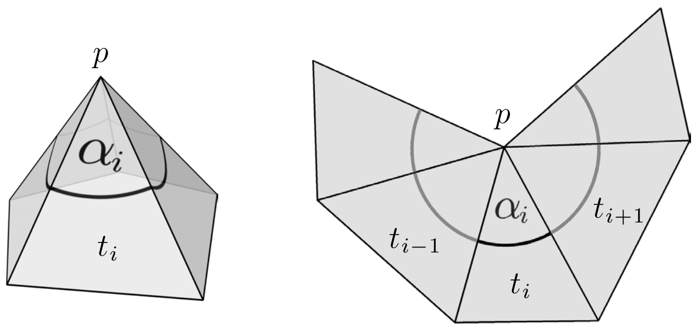


Figure 1.2: The star of a vertex p and its isometric unfolding into the plane. The angle by which vertex angles α_i fail to sum to 2π defines the Gauss curvature at p.

The **shape operator** or **Weingarten map** of a smooth manifold applied to a tangent vector v is defined in differential geometry as $\kappa_v(p) = -D_v N$. It describes the change of the normal vector field in direction of v. A discrete shape operator, defined at an edge e, is defined by Hildebrandt and Polthier [HP04] as

$$S(e) = 2\|\mathbf{e}\| \cos \frac{\alpha_e}{2} \, (\mathbf{e} \times \mathbf{n}_e)(\mathbf{e} \times \mathbf{n}_e)^\intercal$$

where α_e is the dihedral angle at the edge e and $\mathbf{n}_e$ is the unit vector which bisects the adjacent triangle normals. Hildebrandt and Polthier [HP04] also define a vertex-based shape operator, by properly summing up the edge-based operators.

The eigenvectors of the shape operator point in direction of the minimal and maximal normal curvature of the surface and are called the **principal curvature directions**. They will be of interest for the creation of guiding fields in Chapter 5.

1.2 Finite Element Spaces

Our QuadCover algorithm is based on functions which are linear on each triangle. This is motivated by their simple handling and the widespread use of piecewise linear (PL) texture maps.

We consider two spaces of PL functions. The definitions match those in [Dzi88, Pol02, War06] up to the fact that the first two references allow multi-dimensional functions on M. The first finite element space S_h is the space of continuous, PL functions, formally

$$S_h := \left\{ u \in C^0(M) \mid u \text{ is linear in each simplex of } M \right\}.$$

The function f is uniquely determined by its values on the vertices, due to the fact that linear interpolation determines linear functions on each triangle. The vector space S_h is spanned by the **Lagrange basis functions** (φ_p) that are defined at each vertex $p \in \mathsf{V}$. Each basis function φ_p is a linear function on the mesh triangles with the property

$$\varphi_p(p) = 1 \quad \text{and} \quad \varphi_p(q) = 0 \quad \text{for each vertex } q \neq p. \tag{1.3}$$

The other finite element space S_h^* is defined as

$$S_h^* := \left\{ u : M \to \mathbb{R} \;\middle|\; \begin{array}{c} u \text{ is linear in each triangle of } M \text{ and} \\ \text{continuous at all edge midpoints} \end{array} \right\}.$$

The corresponding **Crouzeix-Raviart basis functions** are based on the edges of the mesh and are denoted by (ψ_e) for $e \in \mathsf{E}$. Each basis function is linear on triangles and satisfies

$$\psi_e(e) = 1 \quad \text{and} \quad \psi_e(f) = 0 \text{ for each edge } e \neq f.$$

Here, $\psi_e(f)$ denotes the values of ψ_e at the *midpoint* of edge f. Note that the space of Crouzeix-Raviart elements includes the space spanned by linear Lagrange elements, *i. e.*, $S_h \subset S_h^*$, since

$$\varphi_p = \frac{1}{2} \sum_{e \sim p} \psi_e \tag{1.4}$$

when summing over all edges e incident to p. Unlike Lagrange basis functions, the Crouzeix-Raviart basis functions are generally discontinuous. The functions in the Crouzeix-Raviart space are continuous at the edge midpoints and can be discontinuous on the rest of the edge. Therefore, the linear elements of S_h^* are called *nonconforming*, while the elements of S_h are called *conforming*.

Let $\mathfrak{X}(M)$ denote the space of tangential vector fields on M, which are piecewise constant on each triangle of M. Particular elements of $\mathfrak{X}(M)$ are the gradient fields of the aforementioned PL functions. For example, the gradient of a Lagrange basis function φ_p on triangle $\mathbf{t}$ is given by

$$\nabla \varphi_{p|\mathbf{t}} = \frac{1}{2A_{\mathbf{t}}} \mathbf{J}\mathbf{e},$$

where $A_{\mathbf{t}}$ is the area of triangle t, e is the positively oriented edge of t that does not contain p, and $\mathbf{J}$ is the 90 degrees rotation of tangent vectors in their respective tangent plane, see Figure 1.3, left. (Notice that $\mathbf{J}$ is also well defined on the interior of edges, because each pair of adjacent edges is intrinsically flat.)

Similarly, the gradient of the Crouzeix-Raviart basis function ψ_e based at the midpoint of edge $\mathbf{e}$ (*cf.* Figure 1.3, right) is

$$\nabla \psi_{e|\mathbf{t}} = -\frac{1}{A_{\mathbf{t}}} \mathbf{J}\mathbf{e}.$$

The gradient of a function $u \in S_h$ (and analogously, $u \in S_h^*$) in a triangle $\mathbf{t}$ is then determined by the sum of gradients of the three basis functions contributing to $u_{|\mathbf{t}}$, weighted by the values of u at the vertices.

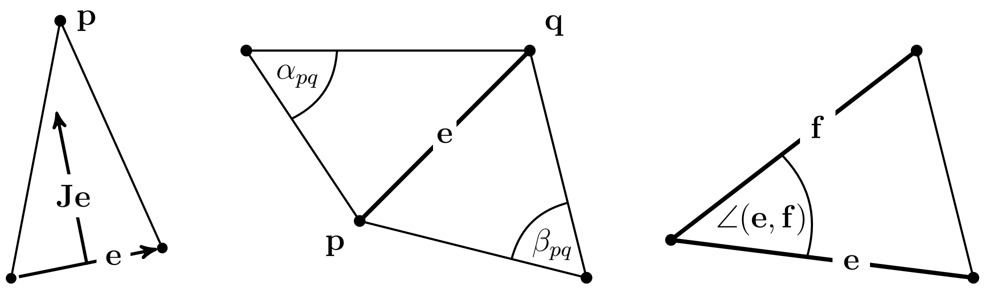


FIGURE 1.3: Labelings of **Je**, α_{pq}, β_{pq} and $\angle(\mathbf{e},\mathbf{f})$.

1.2.1 DIRICHLET ENERGY AND THE LAPLACE OPERATOR

The Dirichlet energy measures how *variable* a function is. It plays a central role due to the fact that its gradient defines the Laplace-Beltrami operator and its minimizers are the harmonic functions.

Definition 1.9 *The* ***Dirichlet energy*** E_D *of a piecewise linear function* $u \in S_h^*$ *(possibly* $u \in S_h$*) is*

$$E_D(u) := \frac{1}{2}\int_M \|\nabla u\|^2 \mathrm{d}A.$$

The well-known cotangent formulas [PP93] are commonly used to explicitly compute the Dirichlet energy. For $u \in S_h$ we have

$$E_D(u) = -\frac{1}{2}\sum_{\{q,p\}\in \mathsf{E}} \mathfrak{L}_{pq}\left(u(p) - u(q)\right)^2$$

where $(\mathfrak{L}_{pq})$ is the **conforming stiffness matrix** of M, with entries

$$\mathfrak{L}_{pq} = \int_M \langle \nabla\varphi_p, \nabla\varphi_q\rangle\,\mathrm{d}A = \begin{cases} -\frac{1}{2}(\cot\alpha_{pq} + \cot\beta_{pq}) & \text{if } p \sim q \\ -\sum_{r\sim p} \mathfrak{L}_{pr} & \text{if } p = q \\ 0 & \text{otherwise.}\end{cases}$$

The angles α_{pq} and β_{pq} are depicted in Figure 1.3. Likewise, if $u \in S_h^*$, then

$$E_D(u) = -\frac{1}{2}\sum_{\substack{e,f\,\in\,\mathsf{E} \\ e\,\sim\, f}} \mathfrak{L}^*_{ef}\left(u(e) - u(f)\right)^2$$

Recall that edges e and f are incident, $e \sim f$, if they share exactly one vertex and that $u(e)$ denotes the value of u at the midpoint of edge e. The corresponding

nonconforming stiffness matrix $(\mathfrak{L}^*_{ef})$ has the following form:

$$\mathfrak{L}^*_{ef} = \int_M \langle \nabla\psi_e, \nabla\psi_f\rangle \,\mathrm{d}A \;=\begin{cases} -2\cot\angle(\mathbf{e},\mathbf{f}) & \text{if } e \sim f \\ -\sum_{g\sim e}\mathfrak{L}^*_{eg} & \text{if } e = f \\ 0 & \text{otherwise} \end{cases} \tag{1.5}$$

where angle $\angle(\mathbf{e},\mathbf{f})$ is measured in the interval $[0,\pi]$ as in Figure 1.3, independently of the orientation of $\mathbf{e}$ and $\mathbf{f}$.

Analogously to the smooth case, the **Laplace-Beltrami operator** can defined as the negative gradient of the Dirichlet energy, and we get again a conforming and a nonconforming version of the Laplace-Beltrami operator, depending on whether it is evaluated at the vertices or the edge midpoints.

Definition 1.10 *Let $u \in S_h$. Then the* ***conforming discrete Laplace-Beltrami operator*** Δ *is defined on the vertices of M as*

$$\Delta u(p) = -\int_M \langle \nabla u, \nabla\varphi_p\rangle \mathrm{d}A.$$

Similarly, for $u \in S^*_h$, the ***nonconforming discrete Laplace-Beltrami operator*** Δ^* *is defined at the edge midpoints as*

$$\Delta^* u(e) = -\int_M \langle \nabla u, \nabla\psi_e\rangle \mathrm{d}A.$$

The Laplace-Beltrami operator has its analogous representation in terms of the stiffness matrix. For $u \in S_h$ we have

$$\Delta u(p) = \sum_{q\sim p} \mathfrak{L}_{pq}\ (u(p) - u(q)),$$

and for $u \in S^*_h$,

$$\Delta^* u(e) = \sum_{f\sim e} \mathfrak{L}^*_{ef}\ (u(e) - u(f)).$$

A conforming PL function $u \in S_h$ is called **harmonic** if it satisfies $\Delta u(p) = 0$ at all vertices $p \in \mathsf{V}$. Analogously, a nonconforming function $v \in S^*_h$ is called harmonic, if $\Delta^* u(e) = 0$ at all edge midpoints.

Unfortunately, the so-defined Laplace-Beltrami operators fail to satisfy the *maximum principle* that is inherent to its smooth counterpart: harmonic functions have no local maxima (or minima) at interior points. There exist several other discretizations of the Laplace-Beltrami operator, but all of them fall short of satisfying all natural properties of the smooth Laplacian operator. As we showed in our *no free lunch* paper [WMKG07], there exist meshes which do not admit any discrete Laplacian operator—defined by its linear action on vertex based functions—that satisfy all of the following four properties:

Symmetry $\mathfrak{L}_{ij} = \mathfrak{L}_{ji}$

Locality $\mathfrak{L}_{ij} = 0$ if v_i and v_j do not share an edge

Linear precision $\Delta u_i = 0$ at each interior vertex v_i if M is a planar mesh and u is a linear function on M, point sampled on the vertices of M

Positive weights $\mathfrak{L}_{ij} \geq 0$ unless $i = j$ and for each i, there exists at least one vertex $j \sim i$ such that $\mathfrak{L}_{ij} > 0$.

The *positive weights* property, which is a sufficient condition to fulfill the maximum principle, is violated by the so called cotan-Laplacian of [PP93] as defined above. However, since the cotan Laplacian possesses the other three desirable properties above, it is widely used in computational geometry.

Matrix Notation

Let $u \in S_h$. Then the coefficients $u_{p_1}, \ldots, u_{p_\mathsf{v}}$ in the basis function representation $u = \sum_{p \in \mathsf{V}} u_p \varphi_p$ are completely determined by the values at the mesh vertices (respectively edge midpoints for $v = \sum_{e \in \mathsf{E}} v_e \psi_e$). Thus it makes sense to express the function coefficients of $u \in S_h$ and $v \in S_h^*$ as vectors $\mathbf{u} = (u_{p_1}, \ldots, u_{p_\mathsf{v}}) \in \mathbb{R}^\mathsf{v}$ and $\mathbf{v} = (v_{e_1}, \ldots, v_{e_\mathsf{e}}) \in \mathbb{R}^\mathsf{e}$, where v and e, as always, denote the number of vertices and edges.

With this convention, the Dirichlet energy and the Laplace-Beltrami operator have particularly simple representations using the stiffness matrices $\mathfrak{L}$ and $\mathfrak{L}^*$:

$$\begin{aligned} E_D(u) &= \tfrac{1}{2}\mathbf{u}^\mathsf{T} \mathfrak{L}\, \mathbf{u}, & \Delta u(p) &= (\mathfrak{L}\, \mathbf{u})_p \\ E_D(v) &= \tfrac{1}{2}\mathbf{v}^\mathsf{T} \mathfrak{L}^* \mathbf{v}, & \Delta^* v(e) &= (\mathfrak{L}^* \mathbf{v})_e \end{aligned}$$

The two stiffness matrices $\mathfrak{L}$ and $\mathfrak{L}^*$ are closely related via the **incidence matrix** $\mathbf{C} \in \{0,1\}^{\mathsf{v} \times \mathsf{e}}$ of the edge graph of M, whose entries $\mathbf{C}_{pe}$ are 1 exactly if $p \in e$:

$$\mathfrak{L} = \tfrac{1}{4}\, \mathbf{C}^\mathsf{T} \mathfrak{L}^*\, \mathbf{C}.$$

1.2.2 Divergence and Curl

Like the Laplace operator, curl and divergence of tangential vector fields are commonly discretized in a *conforming* and a *nonconforming* version.

Definition 1.11 (discrete divergence and curl). *Let $X \in \mathfrak{X}$. The **conforming discrete divergence** of X,* div$X$, *is a function on the vertices of M defined as*

$$\operatorname{div} X(p) := \frac{1}{2} \oint_{\partial \operatorname{star}(p)} g_M(X, \nu)\, \mathrm{d}s.$$

*The **nonconforming discrete divergence** of X, $\operatorname{div}^* X$, is a function on the edges of M defined as*

$$\operatorname{div}^* X(e) := \oint_{\partial \operatorname{star}(e)} g_M(X, \nu) \, \mathrm{d}s.$$

*Here, ν is the outward normal at the boundaries, $\partial \operatorname{star}(p)$ and $\partial \operatorname{star}(e)$. Similarly, the **conforming discrete curl** of X, $\operatorname{curl} X$, is a function on the vertices of M defined as*

$$\operatorname{curl} X(p) := \frac{1}{2} \oint_{\partial \operatorname{star}(p)} g_M(X, \tau) \, \mathrm{d}s,$$

*Likewise, the **nonconforming discrete curl**, $\operatorname{curl}^* X$, is a function on the edges of M defined as*

$$\operatorname{curl}^* X(e) := \oint_{\partial \operatorname{star}(e)} g_M(X, \tau) \, \mathrm{d}s.$$

Here, τ is the unit tangent vector at the boundaries, $\partial \operatorname{star}(p)$ and $\partial \operatorname{star}(e)$.

Note that the definitions also make sense if M has boundary. The integral notation conceals the fact that the discrete divergence and curl can be easily computed by simple sums, since the integrands are piecewise constant, and g_M coincides with the Euclidean metric outside the vertices. Using the notation of Figure 1.4 and a fixed edge orientation, the discrete divergence and curl of interior vertices and edges can be computed by

$$\begin{aligned}
\operatorname{div} X(p) &= -\frac{1}{2} \sum_{t_i \ni p} \langle X_{t_i}, \, \mathbf{J}\mathbf{e}_i \rangle \\
\operatorname{div}^* X(e) &= \langle X_s, \, \mathbf{J}\mathbf{e} \rangle - \langle X_t, \, \mathbf{J}\mathbf{e} \rangle \\
\operatorname{curl} X(p) &= \frac{1}{2} \sum_{t_i \ni p} \langle X_{t_i}, \, \mathbf{e}_i \rangle = -\operatorname{div} \mathbf{J} X(p) \\
\operatorname{curl}^* X(e) &= \langle X_s, \, \mathbf{e} \rangle - \langle X_t, \, \mathbf{e} \rangle = -\operatorname{div}^* \mathbf{J} X(e) \qquad (1.6)
\end{aligned}$$

If p or i happen to lie on the boundary, then the terms which reference missing triangles just vanish.

If the triangles $\mathbf{s}$ and $\mathbf{t}$ at edge $\mathbf{e}$ are flattened into one plane, the formulas of $\operatorname{div}^* X$ and $\operatorname{curl}^* X$ simplify to

$$\begin{aligned}
\operatorname{div}^* X(e) &= \langle X_s - X_t \, , \, \mathbf{J}\mathbf{e} \rangle \\
\operatorname{curl}^* X(e) &= \langle X_s - X_t \, , \, \mathbf{e} \rangle,
\end{aligned}$$

The conforming and nonconforming versions of the curl and divergence are related by the the following averaging property:

$$\operatorname{div} X(p) = \frac{1}{2} \sum_{e \ni p} \operatorname{div}^* X(e), \qquad \operatorname{curl} X(p) = \frac{1}{2} \sum_{e \ni p} \operatorname{curl}^* X(e).$$

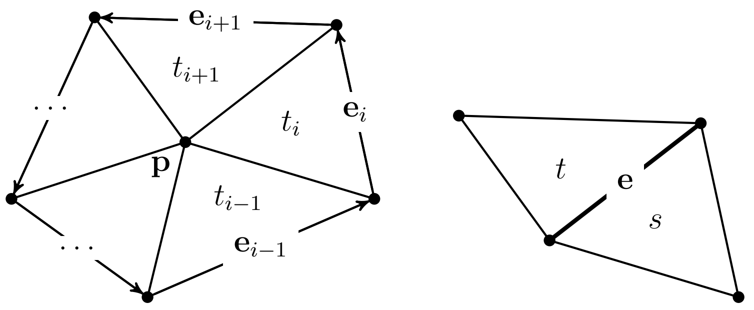


Figure 1.4: Labelings around a vertex **p** and an edge **e**.

When we interpret the values of div $X(p)$ as entries of a vector $\mathbf{div}X \in \mathbb{R}^{\mathsf{v}}$ and interpret $\mathbf{curl}\,X \in \mathbb{R}^{\mathsf{v}}$, $\mathbf{div}^*X \in \mathbb{R}^{\mathsf{e}}$, and $\mathbf{curl}^*X \in \mathbb{R}^{\mathsf{e}}$ as vectors in the same manner, then the averaging property has the following, simple form using the incidence matrix $\mathbf{C}$:

$$\mathbf{div}X = \tfrac{1}{2}\,\mathbf{C}^{\mathsf{T}}\mathbf{div}^*X, \qquad \mathbf{curl}\,X = \tfrac{1}{2}\,\mathbf{C}^{\mathsf{T}}\mathbf{curl}^*X. \tag{1.7}$$

1.2.3 Discrete Hodge Decomposition

A key step of our QuadCover algorithm in Chapter 2 is to find the integrable parts of vector fields via the discrete Hodge decomposition of vector fields, as formulated in its discrete version in [PP03]. We use the notation of Wardetzky [War06]:

Theorem and Definition 1.12 (Discrete Hodge decomposition) *Let M be closed. The space $\mathfrak{X}$ of piecewise constant vector fields on M can be decomposed according to the following (conforming and nonconforming) L^2-orthogonal splittings*

$$\begin{aligned}\mathfrak{X} &= \operatorname{im}\nabla_{|S_h} \oplus \operatorname{im}\mathbf{J}\nabla_{|S_h^*} \oplus \ker \operatorname{curl}^* \cap \ker \operatorname{div} \\ &= \operatorname{im}\mathbf{J}\nabla_{|S_h} \oplus \operatorname{im}\nabla_{|S_h^*} \oplus \ker \operatorname{div}^* \cap \ker \operatorname{curl}\end{aligned}$$

for which wee use the following notation:

$$\begin{aligned}\mathcal{P} &:= \operatorname{im}\nabla_{|S_h} & \mathcal{C} &:= \operatorname{im}\mathbf{J}\nabla_{|S_h^*} & \mathcal{H} &:= \ker \operatorname{curl}^* \cap \ker \operatorname{div} \\ \mathcal{P}^* &:= \operatorname{im}\nabla_{|S_h^*} & \mathcal{C}^* &:= \operatorname{im}\mathbf{J}\nabla_{|S_h} & \mathcal{H}^* &:= \ker \operatorname{div}^* \cap \ker \operatorname{curl}.\end{aligned}$$

In each of the two versions of the Hodge decomposition (conforming and nonconforming), the decomposition identifies three orthogonal vector field classes, compare Figure 1.5. Vector fields $X_{\mathcal{P}}$ in $\mathcal{P}$ possess a continuous potential u such that $X_{\mathcal{P}} = \nabla u$. Vector fields $X_{\mathcal{P}^*} \in \mathcal{P}^*$ have a edge midpoint continuous potential with the analogous property, $\mathcal{P}^* = \nabla v$. Since curl$X_{\mathcal{P}}$ is zero, these fields are free of turbulence, can be viewed as a flow from sources (div$X_{\mathcal{P}}(u) > 0$) to sinks (div$X_{\mathcal{P}}(u) < 0$). Vector fields

$X_{\mathcal{C}} \in \mathcal{C}$ and $X_{\mathcal{C}^*} \in \mathcal{C}^*$ contain the rotational part of the field and may have vortices, but they are divergence free (*i. e.*, they correspond to incompressible flows). These fields are expressed by their co-potentials, $X_{\mathcal{C}} = \mathbf{J}\nabla v$ and $X_{\mathcal{C}^*} = \mathbf{J}\nabla u$. The remaining vector fields in $\mathcal{H}$ and $\mathcal{H}^*$ are called **harmonic vector fields**. They are both divergence free as well as curl free in their respective conforming or non-conforming versions. On closed surfaces, harmonic vector fields correspond to the flow around handles of the surface.

Of the two dual decompositions $\mathcal{P} \oplus \mathcal{C} \oplus \mathcal{P}$ and $\mathcal{P}^* \oplus \mathcal{C}^* \oplus \mathcal{P}^*$ of $\mathfrak{X}$ we will use the former, because the vertex-based, piecewise linear representation of functions accords to the actual usage in geometry processing.

1.2.4 Computation of the Hodge Decomposition

The components of the Hodge decomposition are orthogonal. Thus, the individual contributions of the subspaces $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{H}$ to a vector field $X \in \mathfrak{X}$ can be found by finding the element in the corresponding subspace that is L^2-closest to X. Let $X_{\mathcal{S}}$ be the orthogonal projection of a vector field $X \in \mathfrak{X}$ to a subspace $\mathcal{S} \in \{\mathcal{P}, \mathcal{C}, \mathcal{H}, \mathcal{P}^*, \mathcal{C}^*, \mathcal{H}^*\}$ of $\mathfrak{X}$. Then $X_{\mathcal{S}}$ can be computed via the following minimization problems [PP03]:

$$\begin{aligned}
X_{\mathcal{P}} &= \nabla u, &\quad \text{where} \quad & u = \operatorname*{argmin}_{u \in S_h} \int_M \|X - \nabla u\|^2 \mathrm{d}A \\
X_{\mathcal{C}} &= \mathbf{J}\nabla v, &\quad \text{where} \quad & v = \operatorname*{argmin}_{v \in S_h^*} \int_M \|X - \mathbf{J}\nabla v\|^2 \mathrm{d}A \\
X_{\mathcal{H}} &= X - X_{\mathcal{P}} - X_{\mathcal{C}} && \\
\\
X_{\mathcal{P}^*} &= \nabla v, &\quad \text{where} \quad & v = \operatorname*{argmin}_{v \in S_h^*} \int_M \|X - \nabla v\|^2 \mathrm{d}A \\
X_{\mathcal{C}^*} &= \mathbf{J}\nabla u, &\quad \text{where} \quad & u = \operatorname*{argmin}_{u \in S_h} \int_M \|X - \mathbf{J}\nabla u\|^2 \mathrm{d}A \\
X_{\mathcal{H}^*} &= X - X_{\mathcal{P}^*} - X_{\mathcal{C}^*} &&
\end{aligned}$$

The energies that appear in the formulas above are all quadratic, so their minimizers can be found by solving a linear system. For example, to minimize the energy

$$E_{\mathcal{P}}(u) := \int_M \|X - \nabla u\|^2 \, \mathrm{d}A = \int_M \|X\|^2 + \|\nabla u\|^2 - 2\langle X, \nabla u\rangle \, \mathrm{d}A,$$

we require that its partial derivative with repect to $u(p)$ at any vertex p is zero:

$$\begin{aligned}\frac{d}{du_p}E_{\mathcal{P}}(u) &= \frac{d}{du_p}\Big(E_D(u) - 2\int_M \big\langle X,\, \textstyle\sum_q u_q \nabla\varphi_q\big\rangle \,\mathrm{d}A\Big) \\ &= \Delta u(p) - 2\int_M \big\langle X,\, \nabla\varphi_p\big\rangle\,\mathrm{d}A \\ &\overset{(*)}{=} \Delta u(p) + \mathrm{div}X(p) \overset{!}{=} 0.\end{aligned}$$

The identity $(*)$ follows from the definition of div$X$ and Gauss's theorem,

$$\mathrm{div}X(p) \overset{\text{def}}{=} \frac{1}{2}\oint_{\partial\mathrm{star}(p)} g_M(X,\nu)\,\mathrm{d}s = -\int_{\mathrm{star}(p)} g_M(X,\nabla\varphi_p)\mathrm{d}A.$$

In matrix notation, the potential function u appearing in $E_{\mathcal{P}}(u)$ can be found by solving

$$\mathfrak{L}\,\mathbf{u} = -\mathbf{div}X, \tag{1.8}$$

and the respective potential functions in the minimization problems for $X_{\mathcal{C}}$, $X_{\mathcal{P}^*}$, and $X_{\mathcal{C}^*}$ can be found analogously via

$$\mathfrak{L}^*\,\mathbf{v} = -\mathbf{curl}^*X, \qquad \mathfrak{L}^*\,\mathbf{u} = -\mathbf{div}^*X, \quad \text{and} \quad \mathfrak{L}\,\mathbf{v} = -\mathbf{curl}\,X. \tag{1.9}$$

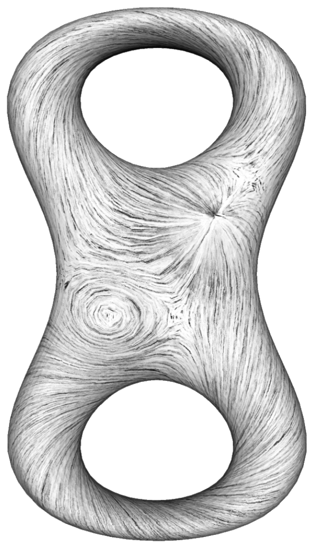
Note that the linear systems of (1.8) and (1.9) do not have unique solutions as the Laplacian operator has constant functions in its kernel. To remove the ambiguity, the solutions can be normalized by requiring

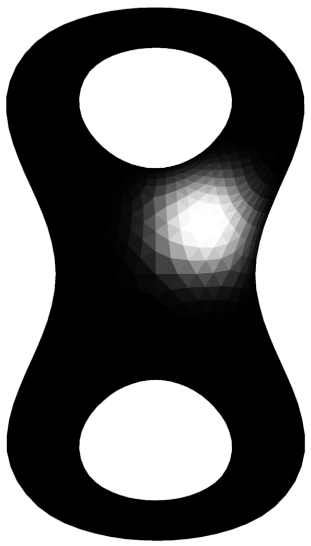
$$\int_M u\;\mathrm{d}A = 0 \quad \text{and} \quad \int_M v\;\mathrm{d}A = 0 \tag{1.10}$$

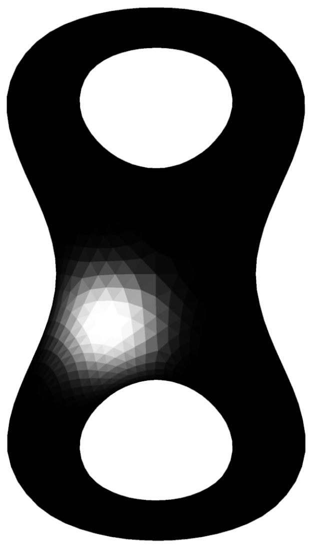
Computationally, the solutions of (1.8) and (1.9) are usually found by computing the sparse Cholesky factorization of $\mathfrak{L}$ and $\mathfrak{L}^*$. The Cholesky factor of a symmetric, positive definite matrix $\mathbf{A}$ is an upper triangular matrix $\mathbf{L}$ satisfying $\mathbf{A} = \mathbf{L}^{\mathsf{T}}\mathbf{L}$. The matrices $\mathfrak{L}$ and $\mathfrak{L}^*$ are indeed symmetric, but only positive semidefinite. This deficit is usually avoided by adding a small positive constant to the diagonal. Once the factor $\mathbf{L}$ is known, the solution of $\mathbf{Ax} = \mathbf{b}$ can be quickly found by consecutively solving $\mathbf{L}^{\mathsf{T}}\mathbf{y} = \mathbf{b}$ and then $\mathbf{Lx} = \mathbf{y}$ row by row. The quick solvability of systems involving the triangular matrix $\mathbf{L}$ makes the Cholesky factorization particularly attractive if the system $\mathbf{Ax} = \mathbf{b}$ needs to be solved for multiple right-hand sides $\mathbf{b}$.

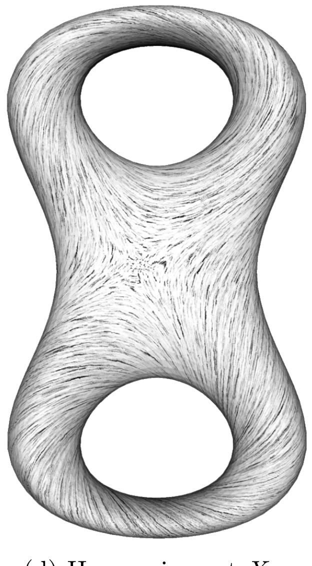
1.2.5 Bases of $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{H}$

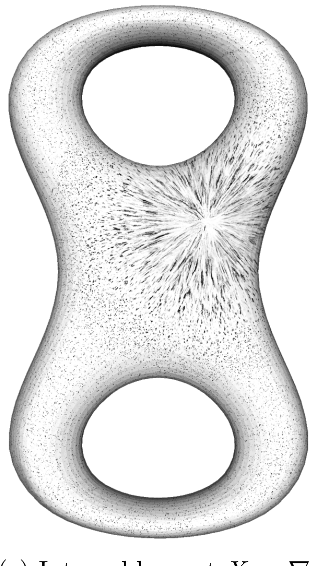
Let M be closed. The two spaces $\mathcal{P}$ and $\mathcal{C}$ have canonical bases, composed of the gradients of the respective basis functions $\varphi_p \in S_h$ and $\psi_e \in S_h^*$. Since $\mathfrak{L}$ and $\mathfrak{L}^*$


(a) Vector field X


(b) Potential u of $X_{\mathcal{P}}$


(c) Co-potential v of $X_{\mathcal{C}}$


(d) Harmonic part $X_{\mathcal{H}}$


(e) Integrable part $X_{\mathcal{C}} = \nabla u$

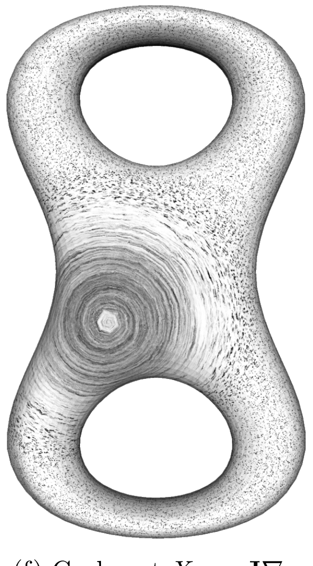

(f) Curl part $X_{\mathcal{P}} = \mathbf{J}\nabla v$

Figure 1.5: Hodge Decomposition: A vector field X is decomposed into its harmonic, potential, and curl part.

have a one-dimensional kernel consisting of the constant functions, we eliminate that additional degree of freedom by normalization via Equation (1.10). Thus, we have $\dim \mathcal{P} = \mathsf{v}-1$ and $\dim \mathcal{C} = \mathsf{e}-1$. Since $\mathfrak{X}$ has two degrees of freedom per triangle, we can see that

$$\dim \mathcal{H} = \dim \mathfrak{X} - \dim \mathcal{P} - \dim \mathcal{C} = 2\mathsf{f} - (\mathsf{v}-1) - (\mathsf{e}-1) = 2\mathsf{g}.$$

The last equality uses Equations (1.1) and (1.2) and the fact that counting the edge-face incidences yields $2\mathsf{e} = 3\mathsf{f}$.

Choosing a basis for the $2\mathsf{g}$-dimensional space $\mathcal{H}$ of harmonic fields is less obvious. To construct a basis for $\mathcal{H}$, we need to recap some more definitions from algebraic topology.

Curves, Homotopy, and Homology

One calls two curves on M with common start and end points **homotopic** if they can be continuously deformed into each other on M. A closed curve, *i. e.*, a curve that starts and ends in the same point, is **contractible** or **null homotopic** if it is homotopic to the constant loop, that is, a point. A closed curve is also called a **cycle** or **loop**. An open (closed) curve is **simple** if it is homeomorphic to a line segment (a circle).

Let p be a fixed point on M. If one calls two loops equivalent whenever they are homotopic, then homotopy defines an equivalence relation on the set of all loops that start and end in p. Together with the concatenation operation, the equivalence classes of loops on M form a group, the **fundamental group** $\pi_1(M, p)$. A **homotopy basis** is a set Γ of loops such that every loop with endpoints in p can be constructed (up to homotopy and orientation) by concatenating paths of Γ. The number of loops in all homotopy bases is always $2\mathsf{g}$.

Following the definition of Erickson and Whittlesey [EW05], a **homology cycle** is a formal linear combination of oriented cycles with coefficients in a ring R, constituting the **homology group** $H_1(M, R)$ of M. The identity element of the homology group is the equivalence class of separating cycles, that is, cycles whose removal disconnects the surface. Two homology cycles are in the same homology class if one can be continuously deformed into the other via a deformation that may include splitting cycles at self-intersection points, merging intersecting pairs of cycles, or adding or deleting separating cycles. We define a **homology basis** for M to be any set of $2\mathsf{g}$ linearly independent cycles. Any homotopy basis is also a homology basis, but not vice versa, since the cycles in a homology basis generally do not have a common point.

Vector Field Integration

Let $X \in \mathfrak{X}(M)$. By the assignment

$$X \longmapsto \omega_X := g_M(X, .\,)$$

there is a unique 1-form ω_X associated with X, which gives rise to integration of vector fields. Let $\gamma : [a, b] \to M$ be a differentiable curve, then the **integral of X along γ** is defined as

$$\int_\gamma \omega_X := \int_a^b g_M\big(X(\gamma(t)), \gamma'(t)\big)\,\mathrm{d}t$$

A vector field is called **integrable**, if there is a function $u \in S_h$ that satisfies $X = \nabla u$ in the interior of all triangles. A vector field is called **locally integrable**, if each point $p \in M$ possesses a (simply connected) neighborhood in which X is integrable.

Lemma 1.13 *The following are equivalent:*

1. $\mathrm{curl}^* X = 0$.
2. X *is locally integrable.*
3. $\int_\gamma \omega_X = 0$ *for all null homotopic curves* γ.

There is a direct consequence of Lemma 1.13. If X is locally integrable, then the value of $\int_\gamma \omega_X$ depends only on the endpoints of γ as long as γ stays within the same homotopy class. Therefore, if M is simply connected, the function

$$f(p) = \int_{p_0}^{p} \omega_X$$

is well defined for a fixed point $p_0 \in M$, and is called the **anti-derivative of ω_X** or **potential function of X**. If $X = \nabla u$ is (globally) integrable, then $f = u$ up to a constant summand due to

$$f(p) = \int_\gamma \omega_{\nabla u} = \int_\gamma g_M(\nabla u, \gamma')\mathrm{d}s = \int_a^b \frac{\mathrm{d}u \circ \gamma(t)}{\mathrm{d}t}\mathrm{d}t = u(p) - u(p_0) \tag{1.11}$$

where γ is a path from p_0 to p. To explicitly compute the integrand on piecewise linear surfaces, we fix the value of u at an arbitrary start vertex, and successively calculate the function values of u at adjacent vertices. When the value of u at vertex p is known, one can use

$$u_q = u_p + < \mathbf{e}, X_t > \tag{1.12}$$

to compute the value of u at an adjacent vertex q, where t is either triangle at the edge $e = (p, q)$. By Equation (1.6), the value of u_q is independent of the choice of t.

The fact that integrals of ω_X always vanish along null homotopic curves raises the question what happens if γ is not null homotopic. The integrals along the non-contractible cycles on M define the periods of X.

Definition 1.14 *The **period** $\mathbf{b}_k^*$ of a locally integrable vector field $X \in \ker \operatorname{curl}^*$ with respect to a cycle $\gamma_k \in \Gamma$ of the homotopy basis is the curve integral*

$$\mathbf{b}_k^*(X) := \int_{\gamma_k} \omega_X.$$

From Lemma 1.13 it is clear why we have to require the local integrability of X for a reasonable definition of periods: Two homotopic curves γ_1, γ_2 with the same basepoint differ by a null homotopic curve that first runs along γ_1 and then backwards along γ_2. Therefore, the periods of X are equal for all curves within the same homotopy class. This makes the definition reasonably well defined. Nevertheless, the Definition 1.14 depends on the choice of the homotopy basis.

Equation (1.11) implies that $\int_\gamma \omega_{\nabla u} = 0$ for all closed curves γ. This leads to the following corollary:

Corollary 1.15 *Periods of (globally) integrable fields vanish.*

Therefore, periods of a locally integrable vector field are solely a characteristic of its harmonic part. Even more, the periods fully characterize a piecewise constant harmonic vector field, resembling the results in the smooth case, compare [Hod52]. For given periods, there is exactly one conforming harmonic vector field. In the following section we will give an explicit construction method, similar to that of Gu et al. [GWY03].

1.2.6 Computing a Base of $\mathcal{H}$

Let $\Gamma = \{\gamma_1, \ \ldots, \ \gamma_{2\mathrm{g}}\}$ be a homotopy basis of M composed of edge-based paths (an explicit construction of a homotopy basis is given by Algorithm 1.16). Choose any fixed orientation on each loop γ_k. This orientation defines a *left side* and *right side* in the local neighborhood of each curve γ_k (assuming no degenerate cases, which can be resolved by local subdivision). We now define for each path γ_k a vector field $\Sigma_k \in \mathfrak{X}$ on the triangles t of M as follows:

$$\Sigma_k(t) = \begin{cases} \sum\limits_{p \in t \cap \gamma_k} \nabla \varphi_p(t) & \text{if } t \text{ is on the left side of } \gamma_k \\ 0 & \text{otherwise,} \end{cases} \tag{1.13}$$

where φ_p is the basis function at vertex p as defined in (1.3). If we were to cut the surface along γ_k and assign the function value 1 to all vertices on the 'left' boundary component created by the cut at γ_k and 0 at all other vertices, then Σ_k would be exactly the gradient of that function. In contrast to the potential function, the vector field Σ_k can be expressed in the standard PL setting without cutting.

It is easy to check that each field Σ_k is curl* free, *i. e.*, that curl$^*\Sigma_k$ vanishes on every edge $e = s \cap t$. If $e \notin \gamma_k$, then Σ_k looks locally like a potential field on $s \cup t$, which are always curl free. Otherwise, Σ_k is perpendicular to e on one side of e and zero on the other, so Σ_k has the same projection to $\mathbf{e}$ from both sides. Thus, the curl on e vanishes in both cases.

We say a cycle γ_k^* is **dual** to a cycle γ_k of the homology basis Γ, if γ_k^* crosses γ_k exactly once, but crosses no other cycle of Γ. The set of dual cycles of Γ also forms a basis of the homology group. The explicit construction of paths with Algorithm 1.16 assures that a dual path does indeed exist for each $\gamma_k \in \Gamma$ (although generally not in Γ). The construction of Σ_k and Equation (1.11) immediately imply

$$\mathbf{b}_j(\Sigma_k) := \int_{\gamma_j^*} \omega_{\Sigma_k} = \delta_{jk},$$

where δ_{jk} is the Kronecker delta.

Periods do not depend on the integrable part of a vector field, so we can subtract the integrable part from Σ_k and get a divergence free (and thus harmonic) field $H_k := \Sigma_k - (\Sigma_k)_{\mathcal{P}}$ with the same periods $\mathbf{b}_j(H_k) = \mathbf{b}_j(\Sigma_k)$. Together, the fields $\{H_1, \ldots, H_{2\mathrm{g}}\}$ form a basis of $\mathcal{H}$, which we call the **standard basis** of harmonic fields (with respect to the homotopy basis $\gamma_1^*, \ldots, \gamma_{2\mathrm{g}}^*$).

To construct the basis fields $H_1, \ldots, H_{2\mathrm{g}}$ explicitly, we compute each harmonic field of the basis via $H_k = \Sigma_k - \nabla u$, where u is the potential of $(\Sigma_k)_{\mathcal{P}}$ found by

$$\mathfrak{L}\mathbf{u} = -\mathbf{div}\,\Sigma_k.$$

Note that the Cholesky factorization of $\mathfrak{L}$ must be computed only once to solve the linear system for all k.

If a harmonic field $H_{\mathbf{b}}$ with desired periods $\mathbf{b}$ is to be constructed, there is no need to compute a basis of $\mathcal{H}$. The field $H_{\mathbf{b}}$ can be directly computed by $H_{\mathbf{b}} = \Sigma_{\mathbf{b}} - \nabla u$ where $\Sigma_{\mathbf{b}} = \sum_{k=1}^{2\mathrm{g}} \mathbf{b}_k \Sigma_k$ and u is the solution of

$$\mathfrak{L}\mathbf{u} = -\mathbf{div}\,\Sigma_b. \tag{1.14}$$

The solution of this system is one of the major steps for QuadCover, and will be referred to again in Section 2.4.2.

1.3 Cut Graphs and Systems of Loops

Our construction of a basis of harmonic functions requires the knowledge of a homotopy basis. We review some more definitions before we give an explicit construction.

Let $G = (V, E)$ be a graph with some node set V and an edge set $E \subset V \times V$. An embedding $\mathcal{X}_M(G)$ of a graph G in M is a map which takes the nodes of G to distinct points on M and maps each edge $e = (v, w)$ to a curve on M which connects the images of v and w. A **cut graph** is an embedding $\mathcal{X}_M(G)$ of a graph in M, such that $M \backslash \mathcal{X}_M(G)$ is a topological disk. A **system of loops** is a cut graph with only one node, *i. e.*, all edges of of the cut graph are loops that start and end at the same node. In particular, each system of loops is a homotopy basis.

1.3.1 Finding a Shortest Cut Graph

An often posed problem is to find the shortest graph with particular properties, where the *length* of a graph is the sum of the lengths of its edges. The edge lengths can be Euclidean lengths (which we assume unless otherwise stated), unit lengths, or any other arbitrary positive weights assigned to the edges.

The problem of finding a shortest cut graph of a triangulated manifold is NP-hard [EHP04]. Erickson and Whittlesey [EW05] describe an algorithm to find a shortest system of loops with a fixed base vertex b, which runs in $O(\mathsf{v}\, log \mathsf{v})$ time (or linear in v, if the genus is regarded as a constant). The algorithm makes use of the edge graph of $G = (\mathsf{V}, \mathsf{E})$ of M, as well as of the dual graph G^* of M, whose vertices are the faces of M with an edge e^* between any pair of faces that share an edge e in G. The proposed algorithm of Erickson and Whittlesey works as following:

Algorithm 1.16 (Shortest System of Loops)

1. *Construct a tree of shortest paths T from the base point b to every other vertex in G (using Dijkstra's algorithm).*
2. *For each edge $e \in G \setminus T$, let $\sigma(e)$ denote the shortest loop that contains e, that is, the edge e plus the direct paths in T from the endpoints of e to the root b.*

   Construct the maximum spanning tree T^* of $(G \setminus T)^*$ with respect to the weight $|\sigma(e)|$ for any dual edge $e^* \in (G \setminus T)^*$ (using Prim's or Kruskal's algorithm).
3. *Output loop $\sigma(e)$ for every edge e that is neither crossed by T nor T^*. Together these loops constitute a shortest system of loops with respect to the base point b.*

In [KNP10], we proposed an extension to this method for a set B of fixed base vertices of the cut graph, as well as for the existence of boundaries: Simply identify all base vertices and all vertices on the boundaries to a single point, and undo the identification after the cut graph is found. This extension reduces the search for the shortest cut graph to finding an optimal number and position of base vertices. In this case, the algorithm outputs a list of paths whose endpoints are in B, that is, each end point is either a singularity or a boundary vertex. Since there may be more than one base point, the output is generally not a system of loops (but still a cut graph).

Independently, Éric Colin de Verdière [dV10] proposed to start the shortest path tree from the base vertex set B, which has the same effect as our identification to a single point. Colin de Verdière succeeded in providing a simple proof for the correctness of the method, but did not mention the fact that surfaces with boundaries can be handled with his method as well.

1.4 Vector Field Indices

Smooth Vector Fields

Vector field indices and their generalizations play a major role for parameterization. We will first have a look at the definition of indices of smooth vector fields, before looking at discrete ones. Let M be a differentiable 2-manifold, possibly with boundary, and let X be a continuous vector field with isolated zeros, and let $p \in M$ be such a zero. In [Ebe01], the index of the vector field X at p is defined as follows.

Definition 1.17 ***Index of a vector field.*** *Let $\varphi : U \to \Omega \subset \mathbb{R}^2$ be a chart around p with $\varphi(p) = 0$, where U is chosen so small, that U contains no further zeros of X. We consider the map*

$$g = D\varphi \circ X \circ \varphi^{-1} : \Omega \longrightarrow \mathbb{R}^2$$

that transfers the vector field from M to Ω via the differential $D\varphi$. Let $S^1_\varepsilon \subset \Omega$ be a small circle of radius ε around 0 in $\mathbb{R}^2$. Since p is the only zero of X on U, we have $g(S^1_\varepsilon) \subset \mathbb{R}^2 \setminus \{0\}$. Let $\tilde{g}$ be the map given by

$$\begin{aligned} \tilde{g} : S^1_\varepsilon &\longrightarrow S^1 \\ x &\longmapsto \frac{g(x)}{\|g(x)\|}. \end{aligned}$$

Let $x \in S^1_\varepsilon$ be a regular point of $\tilde{g}$. Then $T_x\tilde{g} : T_xS^1_\varepsilon \to T_{\tilde{g}(x)}S^1$ is a linear isomorphism. We define $\operatorname{sign} T_x\tilde{g}$ *as $+1$ or -1 as the determinant of the map $T_x\tilde{g}$ is positive or negative. Let $y \in S^1$ be a regular value of $\tilde{g}$. We define*

$$\operatorname{ind}_p(X) := \sum_{x \in \tilde{g}^{-1}(y)} \operatorname{sign}\, T_x\tilde{g}.$$

The number $\operatorname{ind}_p(X)$ *is called the index of the vector field* X *at* p*. One can show that the index is independent of the regular value* $y \in S^1$ *and the chosen chart U, compare [Mil65], §5 and §6.*

Simply speaking, the vector field index at a point p is the signed number of rotations that the direction of a vector field performs when tracing it once around an infinitesimally small circle around p.

Discrete Vector Fields

For our applications on discrete surfaces we need a comparable concept of vector field indices for discrete vector fields. The above definition is not directly transferable to discrete fields. For instance, the fact that all the Gauss curvature of simplicial surfaces is concentrated in the vertices has to be considered for the measure of the index.

Instead of looking at a small smooth circle mapped around a point $p \in M$, we consider the cycle of faces around a vertex. To define the index of a vector field at a vertex, we examine how the vector field changes with respect to a vector that is parallel transported along this cycle.

Following Ray et al. [RVLL08], let $X \in \mathfrak{X}$ a piecewise constant tangential vector field on M. Let s and t be two adjacent triangles (flattened to lie in the same plane, if necessary). We define the (signed) **curvature** θ_{st} of the vector field X along the edge between s and t as the (signed) angle of rotation that brings X_s to X_t. In contrast to the angle function $\angle(X_s, X_t)$ that returns values in the interval $[-\pi, \pi]$, we allow θ_{st} to take any values in $\mathbb{R}$, so θ_{st} also includes information about the number of full rotations, denoted m_{st}, that we had to carry out if we were to interpolate the direction of X_s in triangle s into that of X_t in triangle t.

The number θ_{st} has a unique decomposition

$$\theta_{st} = \angle(X_s, X_t) + 2\pi m_{st}$$

where m_{st} is an integer, provided that in the ambiguous case when $\angle(X_s, X_t) = \pm\pi$, m_{st} and $\angle(X_s, X_t)$ are chosen so that m_{st} has the smaller absolute value. This will assure the consistency condition

$$\theta_{st} = -\theta_{ts} \quad \text{and} \quad m_{st} = -m_{ts}$$

making both θ and m discrete 1-forms on the edges of M.

When we think of a piecewise constant vector field X as a smooth vector field $\mathcal{X}$ which is sampled at the triangle midpoints, then the knowledge of X alone is not enough to determine the vector field indices of $\mathcal{X}$, as no information of the behavior of the vector field between the sample points is given. The angles $\angle(X_s, X_t)$ can be measured, but only with the specification of either m_{st} or θ_{st}, the topology of the vector field between two adjacent triangles is fully defined, and so is the index at the vertices by the following definition. We call the number m_{st} the **matching** between triangles s and t.

Definition 1.18 (Discrete vector field index). *Let* $\{t_1, \dots, t_k\}$ *be the set of faces of* $\operatorname{star}(p)$ *and formally let* $t_{k+1} = t_1$. *The index of a piecewise constant, tangential vector field* $X \in \mathfrak{X}$ *at an interior vertex* p *is given by*

$$\operatorname{ind}_p(X) = \left(K(p) + \sum_{i=1}^{k} \theta_{t_i t_{i+1}}\right) / 2\pi.$$

A vertex p *with* $\operatorname{ind}_p(X) \neq 0$ *is called* singular, *and it is called* regular *otherwise.*

Notice that the index $\operatorname{ind}_p(X)$ is always an integer: if we were to group all terms contributed by the Gauss curvature $K(p)$ and θ by their triangle index, then each triangle t_i would contribute $\big(-\angle(\mathbf{e}, \mathbf{f}) + \angle(\mathbf{e}, X_{t_i}) + \angle(X_{t_i}, \mathbf{f})\big)/2\pi + m_{t_i t_{i+1}} \in \mathbb{Z}$, where e and f are the two edges incident to p and t_i.

Chapter 2

QuadCover Parameterization, Part I

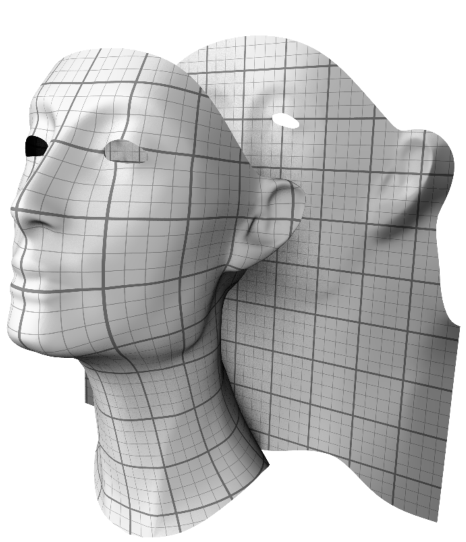


Figure 2.1: A parameterized head model in front of its domain.

2.1 Good Parameterizations

By the **parameterization** of a surface M we understand the process of finding an atlas of M, that is, a set of charts $\varphi_i = (\varphi_i^0, \varphi_i^1) : U_i \to \mathbb{R}^2 \simeq \mathbb{C}$ from subsets $U_i \subset M$ so that the sets U_i cover M. Each parameterization defines a set of **parameter lines** given by $\{p \in M \,|\, \varphi^0(p) \in \mathbb{Z} \vee \varphi^1(p) \in \mathbb{Z}\}$. The triangles of simplicial surfaces give rise to a natural chart layout, where each triangle corresponds to a chart.

This chapter will cover the requirements of a good parameterization from a practical viewpoint and introduce our QuadCover algorithm to find such a parameterization. We have mentioned several applications for mesh parameterization in the introduction, and as different as these applications are, so are the requirements that characterize *good* parameterization. The following non-conclusive list contains the most important requirements with regard to remeshing to quadrilateral meshes, surface texturing, and the creation of a coarse subdivision surface control mesh.

1. **Computability:** Most current mesh processing and rendering hard- and software uses linear interpolation of mesh properties like positions and texture coordinates. Thus, φ should be piecewise linear on M.
2. **Non-degeneracy:** The parameterization φ function should have positive orientation in every triangle, *i. e.*, $\det D\varphi > 0$.
3. **Partition into quads:** For remeshing applications and creation of patch layouts, the parameter lines should partition the surface into quadrilaterals. Formally, the set of parameter lines must be the embedding of a graph whose interior faces have constant degree 4 and whose interior vertices have valence 2 or greater.
4. **Direction control:** In practice, the direction of parameter lines is often prescribed, for example, to follow the principal curvature directions of M, because curvature aligned meshes minimize wrinkles in remeshing applications. This might be a local requirement, for example restricted to sharp bends and surface features.
5. **Density control:** The parameter line density should be adaptive, for example, to create anisotropic meshes, which deliver superior approximations to curved surfaces.
6. **Regularity:** Remeshing and design applications often require the parameterization to be as regular as possible in some metric, for example in the sense that parameter lines
   a) meet perpendicularly,
   b) have equal distance,
   c) are straight (*i. e.*, geodesic).

The fact that many of the listed objectives are conflicting makes surface parameterization hard. In general, the objectives cannot be achieved at the same time. For example, the *regularity* requirement usually conflicts with *partition into quads*, *direction control*, and *density control*. Even more, the requirements 6a, 6b, and 6c are generally pairwise conflicting, so balancing the requirements is always necessary and highly application dependent.

Notice that we require a constant face degree of 4 in Item 3, but we cannot require a constant vertex degree at the same time (unless M happens to have a genus of 1), which can be shown by counting vertices, faces, and edge-face incidences. This implies the existence of irregular vertices, which has a fundamental impact on the structure of the parameterization. In particular we can achieve the regular structure of a planar quadrilateral grid only locally, and we generally will have to deal with irregular vertices that break with the regular structure of the Euclidean plane.

2.2 Related Work

There exists an abundance of different approaches to surface parameterization and, more general, the generation of quad and quad dominant meshes from given triangle meshes.

Conformal Parameterization

Gu and Yau [GY03] were first to construct global quasi-conformal parameterizations of surfaces with arbitrary genus. The resulting parameter lines minimize angle distortion but may have a extremely large length distortion. Kharevych, Springborn, and Schröder [KSS06] used circle patterns to find quasi-conformal parameterizations. In contrast to Gu and Yau, they use cone singularities to increase the flexibility of purely conformal mappings, and can so reduce the length distortion.

The hard problem of finding good positions for cone singularities was tackled by Ben-Chen, Gotsman, and Bunin in [BCGB08]. The authors showed that the length distortion of a conformal metric can be quickly computed to yield prescribed Gauss curvature at the mesh vertices. By the iterative placement of cone vertices, at which the Gauss curvature is concentrated, they could then minimize the length distortion of the conformal parameterization. By leaving the space of piecewise linear functions, Springborn, Schröder, and Pinkall [SSP08] managed to create truly conformal mappings between PL surfaces, and they could also improve the cone placement of Ben-Chen *et al.* by iterative re-positioning.

While global conformal parameterizations minimize angular distortion, the space of conformal parameterizations does not have enough degrees of freedom to allow the local alignment of the parameter lines at given surface features. In other words, conformal parameterizations can excel in the *regularity* category, but they fail the requirements *direction control* and *density control.*

Quadrilateral Patch Layouts

The method of Boier-Martin, Rushmeier, and Jin [BMRJ04] clusters the surface into quadrilateral macro-patches and parameterizes each surface patch separately. Tong, Alliez, Cohen-Steiner, and Desbrun [TACSD06] also use quadrilateral macro-patches, but use global harmonic one-forms surface parameterization functions. Dong *et al.* [DBG$^+$06] compute the Morse-Smale complex of eigenfunctions of the mesh Laplacian to automatically compute a respective patch layout.

The irregular vertices of the quadrilateral meta-layouts become the cone singularities of the parameterization. As detailed in Chapters 3 and 4, the use of cone singularities enlarges the space of harmonic functions on the surface. The presence of cone singularities increases the size of the homology group and thus the dimension of the space of harmonic one-forms on the surface. Still, similar to conformal parameterizations, the approach is constrained by the global nature of harmonic one-forms, and lacks the ability to fulfill requirements *direction control* and *density control*. Huang *et al.* [HZM$^+$08] could eliminate these deficits for the spectral method of Dong *et al.*

Direction Field Driven Parameterization

Alliez et al. [ACSD$^+$03] where first to create quadrangular meshes guided by principal curvature directions. Their approach was extended by Marinov and Kobbelt [MK04], and is based on the integration of curvature lines on the surface. Dong, Kircher, and Garland [DKG05] presented an algorithm which traces isolines in two conjugate harmonic vector fields. Marinov and Kobbelt focus on creating coarse quad-dominant meshes in [MK06] by approximating the surface with very few patches, which are then individually subdivided into quads.

Ray *et al.* [RLL$^+$06] parameterize surfaces of arbitrary genus with periodic potential functions guided by two orthogonal input vector fields. This leads to a continuous parameterization outside the vicinity of cone points on the surface. These singular regions are detected and reparameterized afterwards.

Inspired by this work of Ray *et al.*, our QuadCover parameterization [KNP07] first creates a direction field from the surface's principal curvature directions and then finds a parameterization function whose parameter lines align as much as possible with the given input directions. By regarding the parameterization function as a single-valued function on a branched covering of M, we provide a clean mathematical foundation of how to project direction fields onto the space of parameterization functions via Hodge decomposition.

2.3 Direction Fields

In the list of requirements for parameterizations, Item 4 (*controllable direction*) implies application or user defined direction of parameter lines (possibly only in specific regions of the surface) and Item 6 (*Regularity*) implies perpendicularity of the desired parameter lines. Technically, these two requirements can be described by a set of two mutually perpendicular, tangential vector fields on the surface, whose vectors define the required directions of the surface. Since we assume that this guiding field doesn't need to be more fine grained that the surface mesh itself, we assume all guiding fields to be piecewise constant on the triangles of the mesh. The idea to use such guiding fields first spawned in the Periodic Global Parameterization method [RLL$^+$06], and was well received in the parameterization community.

The term *cross field* commonly describes a set of two tangential, orthogonal vector fields, while the term *n-RoSy fields* (abbreviatory for Rotationally Symmetric fields) refers to a set of n tangential vector fields, which are mutually rotated by $\frac{2\pi}{n}$. Therefore, the first two vectors of a 4-RoSy-field constitute a cross field. In [KNP07], we termed the name *frame field* to denote a set of four tangential vector fields, whose third and fourth vector fields match the first and second field, in reversed direction, compare Section 3.1. Thus, a frame field is a 4-RoSy field without the requirements of perpendicularity and constant length.

The Poincaré-Hopf index theorem states that the sum of vector field indices of a closed surface is equal to the surface's Euler characteristic. One implication of the index theorem is that vector field singularities are unavoidable for surface genuses other than 1. According to Ray *et al.* [RVLL08], a corresponding index theorem holds also for RoSy fields, so we must always reckon on the existence of singularities.

Singularities have the property to make the two vector components of a frame field indistinguishable, in the sense that it is generally possible to 'continuously' trace one frame field component along a closed curve and end up in the other component. Nevertheless, we will assume for a moment the simple case of no singularities and will formulate our QuadCover algorithm here in a simplified version. In this case, a frame field is nothing more than a pair (X, Y) of vector fields. We will continue with frame fields in the next chapter and give more thorough definitions there. The complete QuadCover algorithm which respects the presence of singularities is covered in Chapter 4.

2.4 The QuadCover Algorithm

The QuadCover parameterization picks up the idea of Periodic Global Parameterization [RLL$^+$06] to first find a global guiding field on the surface, and then seek a parameterization function that yields parameter lines in the direction of the guiding field. We first show how we construct a parameterization function from the given guiding field. The generation of suitable frame fields is discussed in Chapter 5. Section 2.4.2 will discuss how the handling of other necessary criteria, such as the continuity of parameter lines, is required to guarantee that they form a *partition into quads* (Requirement 3).

In this chapter, we assume that the guiding field is a frame field (X, Y) without singularities, whose two components X and Y are piecewise constant vector fields on

M. Our task is to find a parameterization function $\varphi = (\varphi^0, \varphi^1) : M \to \mathbb{R}^2$ such that

$$\nabla\varphi^0 = X \qquad \text{and} \qquad \nabla\varphi^1 = Y,$$

if possible. In this case, the parameter lines, *i. e.*, the integer-isolines of φ^0 and φ^1, point in directions perpendicular to X and Y. If we are interested in isolines *along* X and Y, we only have to rotate X and Y by $\frac{\pi}{2}$ in the tangent plane.

2.4.1 Finding an Initial Parameterization

We consider only the first component $X \in \mathfrak{X}$ of the input frame field, because the second component is handled identically. By the Hodge-Helmholtz theorem (Theorem 1.12) X can be split into its mutually orthogonal parts,

$$X = X_{\mathcal{P}} \oplus X_{\mathcal{C}} \oplus X_{\mathcal{H}}$$

and there is a potential function $u \in S_h$ and a co-potential function $v \in S_h^*$ that satisfy

$$X_{\mathcal{P}} = \nabla u \qquad \text{and} \qquad X_{\mathcal{C}} = \mathbf{J}\nabla v.$$

Because the curl part $X_{\mathcal{C}}$ of the vector field is the only part that is non-integrable, the sum $X_{\mathcal{P}} \oplus X_{\mathcal{H}}$ is the best L^2-approximation to the given guiding field *among all vector fields that are gradients of a local parameterization.* One key idea of QuadCover is to drop the curl part of the input field to obtain *the locally integrable vector field that is L^2-closest to the input field.*

Computationally, the first step of the QuadCover is to identify the non-integrable part of X by solving

$$\mathfrak{L}^* \mathbf{v} = -\mathbf{curl}^* X, \tag{2.1}$$

compare Section 1.2.4. If we subtract $\mathbf{J}\nabla v$ from X, then the remainder $\hat{X} := X - \mathbf{J}\nabla v$ is locally integrable, *i. e.*, we can find an integrand φ with $\nabla\varphi = \hat{X}$ on simply connected areas. We take advantage of that fact and cut the surface M along a cut graph Γ (found via Algorithm 1.16) into a topological disk $\mathring{M}$. The local parameterization function then arises by integrating $\hat{X}$ over $\mathring{M}$ via Equation (1.12) and is unique up to an additive constant.

2.4.2 Global Continuity

We now have constructed a continuous parameterization φ on the simply connected surface $\mathring{M}$. When we compare the isolines belonging to φ on both sides of the cut loops of Γ, the lines will generally not join together continuously. They match up if and only

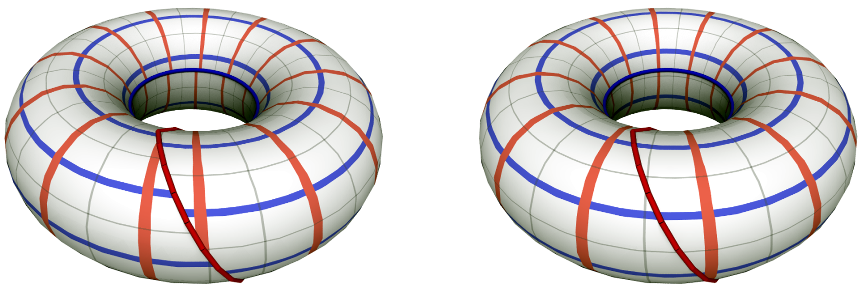


Figure 2.2: Once the curl free field $\hat{X}$ is found by Hodge decomposition, an initial parameterization can be computed by integrating $\hat{X}$. However, parameter lines are discontinuous unless integral periods of the two dual cycles are assured.

if the parameter values at both sides differ by an integer wherever a parameter line meets the cut. In other words, *the parameter lines match if and only if all periods* $\mathbf{b}_k$ *of* $\nabla\varphi$ *with respect to the dual cycles* $\gamma_1^*, \dots, \gamma_{2g}^*$ *are integers* (Figure 2.2).

To achieve that situation, we modify the parameterization φ, but we do it with the smallest possible impact on $\nabla\varphi$ to stay as close to the guiding directions as possible. By Corollary 1.15, adding some potential function of $S_h(M)$ to φ will not alter the periods. Since modifying the curl part would destroy the integrability, this leaves us only with the harmonic functions that we can add to φ to repair the discontinuities.

Let $\mathbf{b} \in \mathbb{R}^{2g}$ with $\mathbf{b}_k = \int_{\gamma_k^*} \omega_{\hat{X}}$ be the period vector of $\hat{X}$ with respect to $\gamma_1^*, \dots, \gamma_{2g}^*$. We have to modify $\hat{X}$ such that all of its periods are integers. A straightforward solution is to round the numbers $\mathbf{b}_1, \dots, \mathbf{b}_{2g}$ to their closest integer value and then find a field $\bar{X}$, that has exactly those periods. We let $\mathbf{d}_k := [\mathbf{b}_k] - \mathbf{b}_k$ denote the offset to the integer $[\mathbf{b}_k]$ that is closest to $\mathbf{b}_k$, and let $H_{\mathbf{d}}$ be the unique harmonic vector field that has periods $\mathbf{d}_1, \dots, \mathbf{d}_{2g}$, found via (1.14). Then the vector field $\bar{X} = \hat{X} + H_{\mathbf{d}}$ has integer periods $[\mathbf{b}_1], \dots, [\mathbf{b}_{2g}]$, and thus its integrand $\bar{\varphi}$ with $\nabla\bar{\varphi} = \bar{X}$ satisfies the required conditions which make the parameter lines match across the cut loops.

Rounding each $\mathbf{b}_k$ to the closest integer is only one reasonable way of rounding. Finding integers such that the norm of the correction field $H_{\mathbf{d}}$ is minimal helps to remain as close as possible to the input directions. However, since this problem is more involved, we postpone it to Chapter 6.3. We end this chapter with a summary of the simplified QuadCover algorithm:

Algorithm 2.1 (simplified QuadCover parameterization)

1. *Construct a guiding frame field X (Chapter 5).*

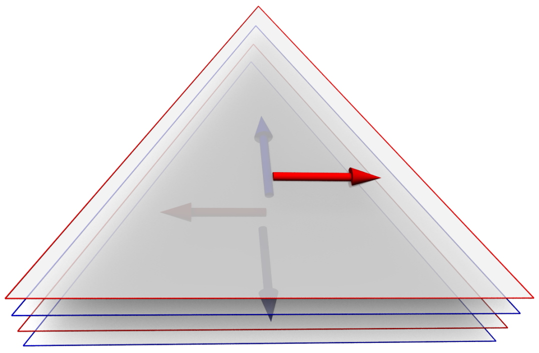
2. *Project X to the space $\mathcal{P} \oplus \mathcal{H}$ of integrable fields by subtracting $X_{\mathcal{C}}$ from X.*

3. *Assure global continuity of parameter lines by adding a harmonic field $H_{\mathbf{d}}$ so that all periods $\mathbf{b}_k + \mathbf{d}_k$ are integers.*

4. *Integration of the resulting field yields the parameterization function φ.*

Chapter 3

Frame Fields and Coverings



3.1 Frame Fields

Vector fields are often given in pairs, or more general, n vector fields are given at each point of the surface. The entirety of vector fields often inherits an additional structure: For example, the eigenvectors of symmetric tensor fields—such as the principal curvature tensors—are always perpendicular wherever they are defined. If two vector fields that could be the eigenvectors of a tensor field were to be designed, additional constraints had to be met to control the relationship between them. The notion of n-way Rotational Symmetry fields (n-RoSy-fields) as defined by Palacios and Zhang [PZ07] specifies exactly these relationships: at each point p of M there exist n unit length vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}$ in the tangent space, and each vector $\mathbf{x}_k$ differs from $\mathbf{x}_0$ by a rotation of $k \cdot \frac{2\pi}{n}$. A 4-RoSy field is often called a *cross field* [HZ00].

RoSy fields have the property of rotational invariance. Thus, a "cross" determined by a single vector, together with a rotation by $\frac{2\pi}{4}$, which generates the three remaining vectors. For QuadCover, we also start with orthogonal frames, but we must loosen the definition of cross fields: to define a Hodge decomposition for cross fields, we cannot hold the orthogonality and unit length requirements. Therefore, our definition of *frame fields* [KNP07] generalizes the definition of cross fields by dropping these requirements.

Let $p \in M$ be a point that does not lie on an edge. A **frame** $(\mathbf{x}, \mathbf{y})$ at a point p is defined by two tangential vectors and the following map R, which adopts the role of the rotation that we had for cross fields. Formally, we define the map

$$\begin{aligned} R : T_pM^2 &\rightarrow T_pM^2 \\ R(\mathbf{x}, \mathbf{y}) &= (-\mathbf{y}, \mathbf{x}). \end{aligned}$$

If $\mathbf{x}$ and $\mathbf{y}$ are orthogonal, then R corresponds to the $\frac{\pi}{2}$-rotation $\mathbf{J}$. The fact that R is a linear automorphism on a real vector space that satisfies

$$R^2 = -\mathrm{id} \tag{3.1}$$

makes R a so-called **complex structure** on T_pM. In fact, the notation is often simpler in complex coordinates. If we define $\mathbf{z} := \mathbf{x} + \boldsymbol{i}\mathbf{y}$, then we have

$$R^k(\mathbf{x}, \mathbf{y}) = \boldsymbol{i}^k \mathbf{z}.$$

Thus, a frame can also be expressed as a complex vector whose real and imaginary parts are tangential vectors of M.

3.1.1 Matchings and Frame Fields

We have seen for vector fields that the discrete set of samples is not enough to compute the index of a vector field, and we need additional information about the transition of a vector towards the vectors in neighboring triangles, and that information is given by the matching.

The transition from vectors to frames introduces a new kind of ambiguity for the comparison of two adjacent objects: It is unclear which of the four individual vectors $X_t, Y_t, -X_t, -Y_t$ must be compared to X_s, when computing angles and differences of two adjacent frames (X_s, Y_s) and (X_t, Y_t). To resolve that ambiguity, we let the matching specify how the individual vectors in the two triangles are paired together, making it essential to determine the field's topology.

Definition 3.1 *A* ***matching*** $m : \mathsf{E} \rightarrow \mathbb{Z}$ *is a map that assigns an integer number to each edge* $e = (s, t)$ *(directed from* s *to* t*) and satisfies* $m_{st} = -m_{ts}$*. The number* m_{st} *specifies that* X_t *is matched to the first component of* $R^{m_{st}}(X_s, Y_s)$*, and* Y_t *is matched to the second component of* $R^{m_{st}}(X_s, Y_s)$ *when evaluating edge based vector field operators at* e*.*

We call the union of two piecewise constant vector fields $X, Y \in \mathfrak{X}$ together with a matching a **frame field** and let $\mathfrak{F}$ denote the space of piecewise constant frame fields on M.

Finally, the curl and divergence of a frame field (X, Y) at an edge $e = (s, t)$ can be computed using the matching. The simplest way is to write them in complex notation, using $Z := X + \boldsymbol{i}Y$:

$$\operatorname{div}^* Z(e) = \langle Z_t,\ \mathbf{Je}\rangle - \langle \boldsymbol{i}^{m_{st}} Z_s,\ \mathbf{Je}\rangle \in \mathbb{C} \tag{3.2}$$

$$\operatorname{curl}^* Z(e) = \langle Z_t,\ \mathbf{e}\rangle - \langle \boldsymbol{i}^{m_{st}} Z_s,\ \mathbf{e}\rangle \in \mathbb{C}. \tag{3.3}$$

3.1.2 Curvature and Indices

Cross Fields

The matching carries two parts of information: m_{st} determines which vectors are matched together, but since $R^4 = \mathrm{id}$ this information depends only on $m_{st} \bmod 4$. For orthogonal frames, m_{st} corresponds to a quarter rotation, so in this case, we interpret m_{st} as the number of quarter rotations to be carried out from triangle s to t.

The **curvature** θ_{st} is the (signed) angle of rotation that rotates X_s to the direction of its matching vector in t, that is,

$$\theta_{st} = \angle(X_s, X_t) + \frac{2\pi m_{st}}{4}. \tag{3.4}$$

The curvature can be interpreted as the sum of three parts: First, a rotation by $\angle(X_s, X_t) \in [-\pi, \pi]$ which brings X_s to X_t, second, a number $m_{st} \bmod 4$ of quarter turns that turn X_t to $X_t^{m_{st}}$, and finally a number of $\lfloor \frac{|m_{st}|}{4} \rfloor$ additional full turns which account for possible vortices. The latter two parts give rise to an index that is solely dependent on the matching:

Definition 3.2 *The* ***index*** ind_p *of a matching m on a simplicial surface M at an interior vertex p is defined as*

$$\operatorname{ind}_p(m) = \frac{1}{4} \sum_{e_{st} \ni p} m_{st},$$

where the edges e_{st} are consistently oriented around p. We call a the index at vertex p ***even*** *if* $\operatorname{ind}_p(m) \in \mathbb{Z} + \frac{1}{2}$, *and* ***odd*** *if* $\operatorname{ind}_p(m) \in \mathbb{Z} \pm \frac{1}{4}$. *If p is a boundary vertex of M, then we set* $\operatorname{ind}_p(m) = 0$.

For consistency, we require as in the vector field case that m_{st} and $\angle(X_s, X_t)$ are to be chosen so that $\angle(X_s, X_t) \in [-\pi, \pi]$ and m_{st} has the smaller absolute value in the ambiguous case when $\angle(X_s, X_t) = \pm\pi$. This will also assure that $\theta_{st} = -\theta_{ts}$. Next, we define the index of cross fields analogously to the index of vector fields:

Definition 3.3 (Cross field index). *The index of a piecewise constant, tangential cross field F at an interior vertex p is given by*

$$\operatorname{ind}_p^{\mathsf{cross}}(F) = \Big(\sum_{\mathbf{e}_{st} \ni p} \theta_{st} + K(p) \Big) / 2\pi.$$

If p is a boundary vertex, then $\operatorname{ind}_p^{\mathsf{cross}}(F)$ *is 0.*

In contrast to the index of vector fields, the frame field index is not an integer, but an integer multiple of $\frac{1}{4}$.

For non-orthogonal frame fields, one could define the curvature and the index as a tuples of two numbers, because the two vector field components could behave independently of each other. Since a complicated, two-componented notion of curvature is not necessary in our applications, we just use above notion of cross field curvature for frame fields by constructing a cross field from an arbitrary frame field. For that purpose, we let

$$(X,Y)^{\perp} := \frac{1}{2}(X - \mathbf{J}Y,\ \mathbf{J}X + Y)$$

be the **orthogonalized frame** of (X,Y). Since the two components $(X,Y)^{\perp}$ are perpendicular, we only need to normalize it to make $(X,Y)^{\perp}$ a cross field. We then define the index of a frame field as the index of the corresponding cross field:

$$\operatorname{ind}_p^{\mathsf{cross}}(X,Y) := \operatorname{ind}_p^{\mathsf{cross}}(X,Y)^{\perp}.$$

3.2 Covering Spaces

3.2.1 Introduction

In his study of complex functions, Bernhard Riemann regarded multi-valued functions on surfaces as single-valued functions on multi-layered surfaces, and so established the study of branched covering surfaces. Our interest in covering surfaces is caused by the fact that parameterization also creates a multi-valued coordinate function on a surface.

We follow Riemann's idea analogously to the description of Stillwell [Sti80]. A complex function $w(z) = z^2$ on the unit disk can be viewed as map of the disk onto itself. The map is not one-to-one, but in a natural sense, $w(z) = z^2$ maps the disk *twice* onto itself except at 0, since any other point of the w-disk is the square of two distinct values $+\sqrt{w}$ and $-\sqrt{w}$. In fact, if we were to divide the z-disk into two half-disks (for example, at the real axis), then squaring would map both half-disks onto the whole w-disk, as shown in the left and middle image in Figure 3.1.

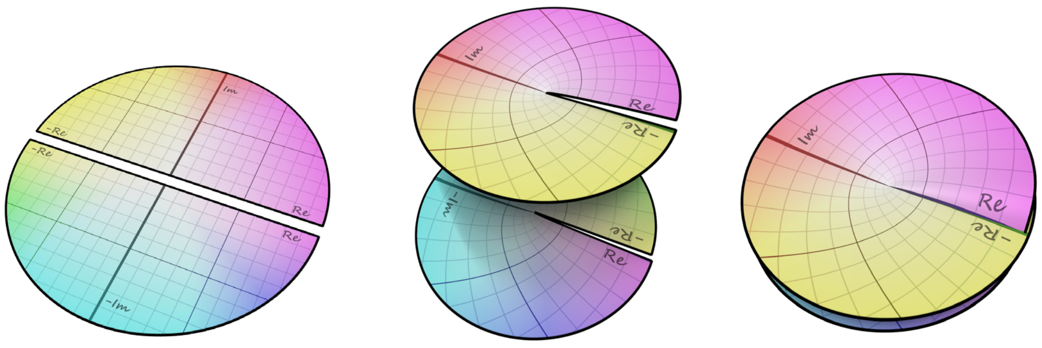


Figure 3.1: *Left:* The z-disk, slit along the real axis. *Middle:* Squaring the half-disks results in two disks with a cut from 0 to 1. *Right:* The two-layered w-disk, after re-identifying those points that were slit apart. The boundary of each neighborhood of the branch point at 0 (in particular, the boundary of the w-disk itself) is composed of two circuits around 0.

Deforming the z-disk so that each point lays above its image on the w-disk results in what we call a 2-*sheeted cover* of the w-disk with branch point 0. The cover of the w-disk can be seen as two disks, each slit along the line segment $[0, 1]$ and identified at the edges labeled with *Re* and *-Re* as shown in Figure 3.1. The two disks are the *sheets*. A small circuit around the origin has the property that it is not closed: it starts on one sheet and ends on the other. Nevertheless the point 0 has a disk neighborhood whose boundary is formed by joining *two* circuits around the branch point (Figure 3.1). Thus, the covering surface is a genuine manifold from the topological point of view, even though this fact is obscured, because our intuition often tries to identify the surface points at the line of self-intersection.

The “two-valued function” $z = \sqrt{w}$ can be viewed as a single-valued function if its domain is taken to be the covering surface instead of the w-disk, which is the general purpose of Riemann surfaces in function theory: to provide domains on which all algebraic functions become single-valued.

In contrast to classic Riemannian geometry, we do not consider algebraic or holomorphic functions, but we are interested in the covering surface’s topology. Surprisingly, topologists seem to generally avoid branched coverings and stick to the unbranched case. To put it in William Massey’s words [Mas64]: “In general, there does not seem to be much known about branched covering spaces”, which he observed in 1967, more than a hundred years after Riemann’s publication. Even though Massey’s observation is dated itself, the absence of branched coverings from topology textbooks seems not to have changed. We use a topological definition of branched covering spaces found in Piękosz [Pię96] that does not rely on the Riemannian structure on the surfaces (but nevertheless shares the same concept).

Definition 3.4 (Branched covering). *Let M be a topological space. A topological space $\widetilde{M}$ together with a map $\pi : \widetilde{M} \to M$ is called an **(n-sheeted) covering** of M if the following property holds: For each point $p \in M$, there exists a neighborhood U_p whose preimage $\pi^{-1}(U_p)$ is the union of exactly n disjoint topological disks.*

*A pair $\widetilde{M}$ and $\pi : \widetilde{M} \to M$ is called a **branched covering** if there exists a finite set of points $P = \{p_1, \dots p_m\} \subset M$ such that the set $\pi^{-1}(P)$ is discrete and the restriction of the map π to $M \setminus P$ is a covering. A point $p \in P$ for which the cardinality of $\pi^{-1}(p_i)$ is smaller than n is called a **branch point**.*

Riemann's concept to investigate multi-valued objects on single-layered surfaces as single-valued objects on multi-layered surfaces can be applied to vector fields as well. Here, frame fields are our multi-valued objects, and they can be identified with vector fields on covering surfaces by the following construction.

3.2.2 A Covering Surface for QuadCover

As always, let M be a piecewise linear surface, and $F \in \mathfrak{F}$, together with its matching m, a frame field on M. According to Riemann's idea, we identify the four-valued frame field $F = (X, Y, -X, -Y)$ on M with a single-valued vector field on a four sheeted covering $\widetilde{M}$ over M, where the four components of $X, Y, -X, -Y$ are spread on the four layers of M.

To define $\widetilde{M}$, we first specify its faces and then their adjacency relations, *i. e.*, the edges of $\widetilde{M}$, in dependence of the matching m. The vertices of $\widetilde{M}$ are then uniquely determined by that construction. Figure 3.2 shows the idea of the construction, which we will now formally describe.

Faces

The face set $\tilde{\mathsf{F}}$ of $\widetilde{M}$ contains four copies of each triangle in M, that is, if the face set of M is $\mathsf{F} = \{t_1, t_2, \dots, t_{\mathsf{f}}\}$, then

$$\tilde{\mathsf{F}} = \{\tilde{t}_1^0, \tilde{t}_1^1, \tilde{t}_1^2, \tilde{t}_1^3, \;\; \tilde{t}_2^0, \; \dots, \tilde{t}_{\mathsf{f}\text{-}1}^3, \;\; \tilde{t}_{\mathsf{f}}^0, \tilde{t}_{\mathsf{f}}^1, \tilde{t}_{\mathsf{f}}^2, \tilde{t}_{\mathsf{f}}^3\},$$

The upper index $l = 0, 1, 2, 3$ of a triangle $\tilde{t}^l$ is the index of the *sheet* or *layer* of the triangle. To simplify the notation when specifying the sheet number, we formally define

$$\tilde{t}^l := \tilde{t}^{l \bmod 4} \quad \text{for all } l \in \mathbb{Z},$$

so that we can omit the 'mod 4' in the layer notation. For the same reason we let the upper indices start with 0, in contrast to the indices of vertices, edges, and triangles, which start at 1.

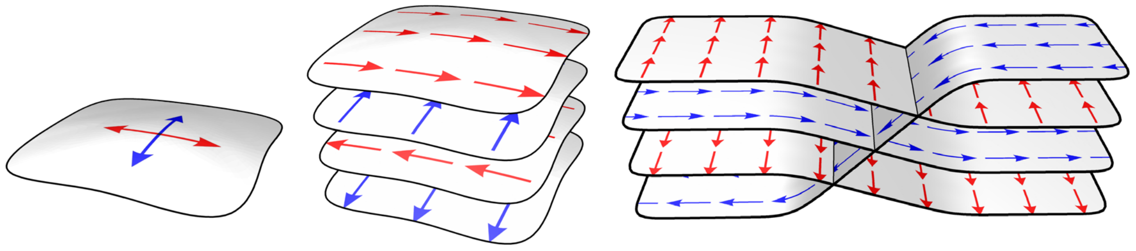


Figure 3.2: Construction of a covering surface: Each face (left) is copied four times (center). Copies that arose from adjacent elements are then glued together according to the matching. The vertex set of the covering is then unique and also the frame field can be spread uniquely to a vector field on the covering surface.

Edges

For each pair of adjacent triangles t_i and t_j in M and $k \in \{0,1,2,3\}$, we let the triangles $\tilde{t}_i^k$ and $\tilde{t}_j^{k+m_{ij}}$ be adjacent.

For formal reasons we assign a layer r to the edge $(\tilde{t}_i^k, \tilde{t}_j^l)$ so that r equals the layer of the triangle with the smaller index,

$$r = \begin{cases} k & \text{if } i < j \\ l & \text{if } i > j. \end{cases} \tag{3.5}$$

With that notation, each edge $e \in M$ gives rise to four well defined edges $\tilde{e}^0, \dots, \tilde{e}^3 \in \widetilde{M}$ and it is assured that $(\tilde{t}_i^k, \tilde{t}_j^l)$ and its reversal $(\tilde{t}_j^l, \tilde{t}_i^k)$ are in the same layer.

Vertices

The vertex set $\tilde{\mathsf{V}}$ of $\widetilde{M}$ is less obvious. Recall from Definition 3.2 that the index $\operatorname{ind}_p(m)$ is one fourth of the sum of matchings around an interior vertex p (which we called *layer shift* in our QuadCover paper).

If $t_0, \dots, t_{d-1}, t_d = t_0$ is a closed cycle of pairwise adjacent triangles around p, then the sequence of layers

$$l_0 = 0, \quad l_1 = m_{t_0 t_1}, \quad l_2 = l_1 + m_{t_1 t_2}, \quad \dots, \quad l_d = \textstyle\sum_{i=0}^{d-1} m_{t_i t_{i+1}} = 4 \operatorname{ind}_p(m)$$

for which the triangles $\tilde{t}_{i-1}^{l_{i-1}}$ and $\tilde{t}_i^{l_i}$ in $\widetilde{M}$ are adjacent is uniquely defined, and $4 \operatorname{ind}_p(m)$ is exactly the layer of the final element. Let us take a look at cases that can arise:

$\operatorname{ind}_p(m) \in \mathbb{Z}$: The cycle of faces

$$\tilde{t}_0^k, \ \tilde{t}_1^{k+m_{01}}, \ \dots, \ \tilde{t}_d^{k+\operatorname{ind}_p(m)}$$

in $\widetilde{M}$ is closed for any $k \in \{0,1,2,3\}$, since the first element equals the last one.

ind_p is even: $\tilde{t}_0^k,\ \tilde{t}_1^{k+m_{01}},\ \ldots,\ \tilde{t}_d^{k+4\,\mathrm{ind}_p(m)}$ is not a closed cycle since we end up with an offset of two layers, but

$$\begin{array}{ll} \tilde{t}_0^0,\ \tilde{t}_1^{l_1},\ \ldots,\ \tilde{t}_d^{l_d},\ \tilde{t}_1^{l_d+l_1},\ \ldots,\ \tilde{t}_d^{2l_d} = \tilde{t}_d^0 & \text{and} \\ \tilde{t}_0^1,\ \tilde{t}_1^{l_1+1},\ \ldots,\ \tilde{t}_d^{l_d+1},\ \tilde{t}_1^{l_d+l_1+1},\ \ldots,\ \tilde{t}_d^{2l_d+1} = \tilde{t}_d^1 & \end{array}$$

are closed cycles of $2d$ faces in $\widetilde{M}$. Also, these two cycles are the only cycles around p, since the ones starting at $\tilde{t}_0^2$ and $\tilde{t}_0^3$ coincide with the two cycles above: the above cycles contain $\tilde{t}_0^2 = \tilde{t}_0^{l_d}$ and $\tilde{t}_0^3 = \tilde{t}_0^{l_d+1}$ already.

ind_p is odd: The cycle of faces starting at $\tilde{t}_0^0$ has $4d$ elements and contains all triangles $\tilde{t}_i^k$ for all $k = 0, 1, 2, 3$ and $i = 0, 1, \ldots, d-1$.

The study of closed cycles on $\widetilde{M}$ suggests how the vertex set $\tilde{\mathsf{V}}$ of $\widetilde{M}$ must look like. The definition of simplicial surfaces requires that every vertex is surrounded by a loop of pairwise adjacent faces, so if $\widetilde{M}$ is to be a simplicial surface, $\tilde{\mathsf{V}}$ must contain exactly one vertex for each of the mentioned cycle of faces in $\widetilde{M}$. Thus, each vertex $p \in M$ with odd index contributes one vertex to $\widetilde{M}$, each vertex with even index contributes two vertices, and all other vertices contribute four vertices each to $\widetilde{M}$.

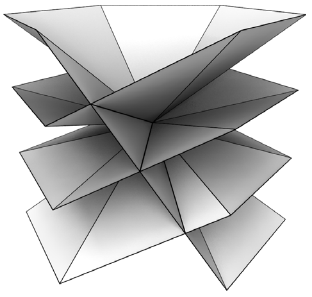


Figure 3.3: Face adjacencies define the cycle of faces around each branch point.

To assign a layer to vertices, we denote the vertex of $\widetilde{M}$ that is incident to the cycle $\tilde{t}_0^k, \tilde{t}_1^{k+m_{01}}, \ldots, \tilde{t}_0^k$ with $\tilde{p}^k$. Note that this notation depends on the choice of the start triangle of the cycle and that it is not unique. For example, $\tilde{p}^0$ and $\tilde{p}^2$ denote the same vertex if the index is even, since $\tilde{t}_0^0$ and $\tilde{t}_0^2$ then lie in the same cycle. The simplest way to resolve the dependence of the start triangle is to fix triangle t_0 as being the triangle at vertex p with the lowest index in a global indexing of all triangles of M.

Remark

The above construction conceals that the four layers of the covering surface coincide geometrically, so a proper induction of a topology from the surrounding space is not possible. To be technically precise, we would have to specify $\widetilde{M}$ via its abstract simplicial complex $\widetilde{\mathfrak{M}}$, consisting of the abstract vertices $\widetilde{\mathfrak{V}}$, edges $\widetilde{\mathfrak{E}}$, and faces $\widetilde{\mathfrak{F}}$, that are constructed analogously to the sets V, E, and F. Finally, we would have to assign the geometric position that belongs to $\mathfrak{v}$ to each vertex $\tilde{\mathfrak{v}}^l$, which implies that edges and triangles of $\widetilde{M}$ would also share the same geometry as their counterparts im M.

If $\widetilde{\mathfrak{M}}$ is constructed that way, the following properties of $\widetilde{\mathfrak{M}}$ follow directly from the construction of $\widetilde{\mathfrak{V}}, \widetilde{\mathfrak{E}}$, and $\widetilde{\mathfrak{F}}$:

Lemma 3.5

1. *$\widetilde{\mathfrak{M}}$ is an abstract simplicial surface.*

2. *The assignment of vertex positions $\tilde{\mathfrak{v}}_i^k \mapsto p(\mathfrak{v}_i)$ to every vertex $\tilde{\mathfrak{v}}_i^k \in \widetilde{\mathfrak{V}}$ makes $\widetilde{\mathfrak{M}}$ a triangle mesh, denoted by $\widetilde{M}$ (where $p : \mathfrak{V} \to \mathbb{R}^m$ is the assignment of vertex positions to the vertices of M). $\widetilde{M}$ is a triangle mesh in the spirit of Definition 1.6, but it is not a simplicial complex due to its self-intersections.*

3. *Let $\widetilde{K}$ and K be the standard realizations of $\widetilde{\mathfrak{M}}$ and $\mathfrak{M}$, respectively (see Definition 1.5). Because $|\widetilde{\mathfrak{V}}| \geq |\mathfrak{V}|$, there is a unique linear map $\pi : \widetilde{K} \to K$ that maps the position of vertex $\tilde{\mathfrak{v}}_i^k$ in $\widetilde{K}$ to the position of vertex $\mathfrak{v}$ in K for every $\tilde{\mathfrak{v}}_i^k \in \widetilde{\mathfrak{M}}$. The pair $(\widetilde{K}, \pi)$ is a 4-sheeted branched covering of K according to Definition 3.4, and the (positions of) vertices $\{\tilde{\mathfrak{v}}_i^k \in \widetilde{\mathfrak{V}} \,|\, \mathrm{ind}_{\mathfrak{v}}(m) \notin \mathbb{Z}\}$ are branch points of $(\widetilde{K}, \pi)$.*

4. *Let b_{even} and b_{odd} be the numbers branch points with an even and odd index, respectively. If $\mathsf{b}_{\text{odd}} > 0$, then $\widetilde{\mathfrak{M}}$ is connected and the genus $\tilde{\mathsf{g}}$ of $\widetilde{\mathfrak{M}}$ satisfies*

$$\tilde{\mathsf{g}} \;=\; 4\mathsf{g} + \tfrac{1}{2}\,\mathsf{b}_{\text{even}} + \tfrac{3}{2}\,\mathsf{b}_{\text{odd}} - 3.$$

   This is a special case of the Riemann-Hurwitz formula and can easily be verified using the definitions of the genus and Euler characteristic in Equations (1.1) and (1.2). In the case that $\mathsf{b}_{\text{odd}} = 0$ and $\mathsf{b}_{\text{even}} > 0$, the surface $\widetilde{\mathfrak{M}}$ decomposes into two connected components, each with a genus of $2\mathsf{g} + \frac{1}{2}\mathsf{b}_{\text{even}} - 1$. If $\mathsf{b}_{\text{odd}} = \mathsf{b}_{\text{even}} = 0$, then $\widetilde{\mathfrak{M}}$ decomposes into four connected components of genus g.

3.2.3 Calculus on $\widetilde{M}$

So far we constructed a covering surface $\widetilde{M}$ from M, and it only remains to lift a multi-valued frame field $(X, Y) \in \mathfrak{F}$ or a function $(\varphi_0, \varphi_1) : M \to \mathbb{R}^2$ from M to the layers of $\widetilde{M}$.

Since the description is much simpler in complex notation, let us define $Z = X + \boldsymbol{i}Y$ (or $Z = \varphi_0 + \boldsymbol{i}\varphi_1$ if we are looking at functions). We now define a real vector field (or real function) $\widetilde{Z}$ on $\widetilde{M}$ by setting

$$\widetilde{Z}(\tilde{p}^l) := \mathfrak{Re}(\boldsymbol{i}^{-l} Z(p))$$

for every $p \in \widetilde{M}$, where l is the layer of point $p \in \widetilde{M}$. The correspondence is one-to-one, and so this assignment has an inverse via

$$Z(p) = \widetilde{Z}(\tilde{p}^0) + \boldsymbol{i}\widetilde{Z}(\tilde{p}^1). \tag{3.6}$$

To compute a Hodge decomposition on $\widetilde{M}$, we need the stiffness matrix of $\widetilde{M}$. The non-conforming stiffness matrix of $\widetilde{M}$ is according to Equation (1.5)

$$\widetilde{\mathfrak{L}}^*_{i^k j^l} = \int_M \langle \nabla \widetilde{\Psi}^k_i, \nabla \widetilde{\Psi}^l_j \rangle \,\mathrm{d}A = \begin{cases} -2\cot\angle(e^k_i, e^l_j) & \text{if } e^k_i \sim e^l_j \\ -\sum\limits_{e^m_h \sim e^k_i} \widetilde{\mathfrak{L}}^*_{i^k h^m} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

where $\widetilde{\Psi}^k_i$ is the Crouzeix-Raviart basis function at edge $\tilde{e}^k_i$. The stiffness matrix is a block matrix containing $\mathsf{e} \times \mathsf{e}$ blocks of size 4×4. The non-zero values of a block at position $\{i^0, \dots, i^3\} \times \{j^0, \dots, j^3\}$ correspond up to the sign to $\mathfrak{L}^*_{ij}$, since the geometry of the individual triangles of $\widetilde{M}$ is the same as those of M. The sign and position of the matrix entries within each block depends on how the individual triangles are connected in the covering surface $\widetilde{M}$.

The stiffness matrix of $\widetilde{M}$ can be written more elegantly as a complex matrix $\bar{\mathfrak{L}}^* \in \mathbb{C}^{\mathsf{e}\times\mathsf{e}}$. Besides having a much cleaner notation, the complex matrices are advantageous for Cholesky updates, as we discuss later in Section 6.2.1. If $\mathfrak{L}^*$ is the real-valued, non-conforming stiffness matrix of M, then the Hermitian matrix

$$\bar{\mathfrak{L}}^*_{ij} = \begin{cases} \boldsymbol{i}^l \mathfrak{L}^*_{ij} & \text{if } \tilde{e}^l_i \sim \tilde{e}^0_j \\ \mathfrak{L}^*_{ii} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

acts on complex valued functions $\bar{u}$ on M in the same way as the block matrix of (3.7) acts on symmetric functions $\tilde{u}$ on $\widetilde{M}$. As an example, the Dirichlet energy of a symmetric function $\tilde{u} \in S_h(\widetilde{M})$ is

$$E_D(\tilde{u}) = \frac{1}{2} \tilde{\mathbf{u}}^\intercal \,\widetilde{\mathfrak{L}}^*\, \tilde{\mathbf{u}} = \bar{\mathbf{u}}^\intercal \,\bar{\mathfrak{L}}^*\, \bar{\mathbf{u}} \tag{3.9}$$

where we have $\bar{\mathbf{u}}_i = \tilde{\mathbf{u}}_{4i} + \boldsymbol{i}\,\tilde{\mathbf{u}}_{4i+1}$ when written in vector notation. Since $\bar{\mathfrak{L}}^*$ is Hermitian, its associated quadratic form $\bar{\mathbf{u}} \mapsto \bar{\mathbf{u}}^\intercal \,\bar{\mathfrak{L}}^*\, \bar{\mathbf{u}}$ is always real. The factor $\frac{1}{2}$ stems from the fact that we skipped layers 2 and 3 in the complex notation.

The adjacency $\tilde{e}^l_i \sim \tilde{e}^0_j$ in Equation 3.8 implies that there is some triangle $\tilde{t}^k$ in $\widetilde{M}$ that contains edges $\tilde{e}^l_i$ and $\tilde{e}^0_j$. Note that by our definition of the layers of edges, (Equation 3.5), l also depends on the indexing of the triangles containing $\tilde{e}^l_i$ and $\tilde{e}^0_j$. Thus, the layer k is not necessarily equal to 0 or l.

The conforming stiffness matrix is constructed equivalently: For each pair of vertices v_p, v_q, there is a unique layer l so that $\tilde{v}^l_p$ is adjacent to $\tilde{v}^0_q$. This layer determines the entries $\overline{\mathfrak{L}}_{pq}$ of the conforming complex stiffness matrix of $\widetilde{M}$,

$$\overline{\mathfrak{L}}_{pq} = \begin{cases} 0 & \text{if } \mathfrak{L}_{pq} = 0 \text{ or } p \text{ or } q \text{ is a branch point} \\ \boldsymbol{i}^l \mathfrak{L}_{pq} & \text{if } \tilde{v}_p^l \sim \tilde{v}_q^0 \\ \mathfrak{L}_{pp} & \text{if } p = q. \end{cases}$$

Remember that the layer of the layers of the vertices of $\widetilde{M}$, and thus also l, depends on the global indexing of all elements touching v_p or v_q.

The problem of ambiguous layers at branch points is dealt with by setting rows and columns corresponding to branch points to zero. Since a branch vertex v_p always appears in multiple layers, any symmetric function at p must satisfy $\tilde{u}_p^0 = \tilde{u}_p^2$ as well as the symmetry condition $\tilde{u}_p^0 = -\tilde{u}_p^2$ at the same time, so $\tilde{u}_p^k$ always vanishes in all layers k. Therefore, omitting the stiffness matrix values at branch points is reasonable. Reasonable non-zero function values at branch points can be realized by setting the function value along a symmetric cycle through the branch points as explained in Section 4.2.

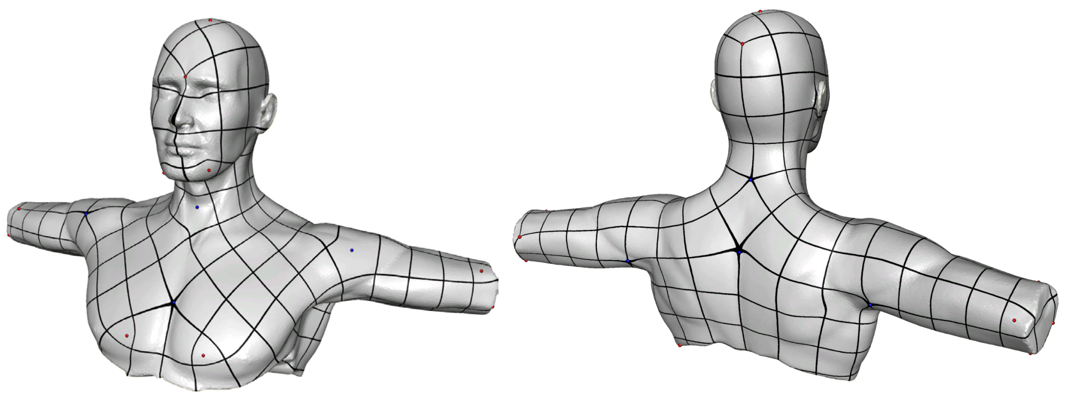
Hodge Decomposition

Knowing the stiffness matrix of $\widetilde{M}$, the Hodge decomposition $\widetilde{\mathfrak{X}} = \widetilde{\mathcal{P}} + \widetilde{\mathcal{C}} + \widetilde{\mathcal{H}}$ of frame fields on $\widetilde{M}$ is straight forward using the equations of Section 1.2.4. The only difference is that we now use the complex valued curl and divergence functions and stiffness matrices $\overline{\mathfrak{L}}$ and $\overline{\mathfrak{L}}^*$ as described above.

As in the vector field case, finding a basis for the spaces $\widetilde{\mathcal{P}}$ and $\widetilde{\mathcal{C}}$ is trivial, but specifying a basis for the space $\widetilde{\mathcal{H}}$ of harmonic functions on $\widetilde{M}$ is more involved. We have already described how to find a homotopy basis that is needed for the construction of bases of $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ in Section 1.3. In Section 4.2 we will show how the respective bases look on the covering surface.

Chapter 4

QuadCover Parameterization, Part II



4.1 QuadCover on Coverings

In Chapter 2 we assumed that we had given a frame field $F = (X, Y)$ without considering the matching. We now consider the more general case of QuadCover, *i. e.*, input fields F endowed with a matching and the existence of singularities.

The simplified QuadCover algorithm in Chapter 2 came in two stages: Parameterization on a simply connected integrable field that is close to the input directions in the first stage, and guaranteeing continuous parameter lines in the second step.

Following the idea of Chapter 3, we regard the frame field $F = (X, Y)$ on the input surface M as a vector field on the four-sheeted covering surface $\widetilde{M}$. The four vectors $X, Y, -X, -Y$ are spread to the four layers of $\widetilde{M}$, giving rise to a vector field $\widetilde{X}$ on $\widetilde{M}$.

In the first step we seek a parameterization function $\widetilde{\varphi} : \widetilde{M} \to \mathbb{R}^2 \simeq \mathbb{C}$ so that $\nabla \widetilde{\varphi}_i^l$ is close to $\widetilde{X}_i^l$ in each triangle t_i and each layer l. Following the steps of Chapter 2, we solve the linear system

$$\widetilde{\mathfrak{L}}^* \tilde{\mathbf{v}} = -\mathbf{curl}^* \widetilde{X},$$

analogously to Equation 2.1, this time using the vector field and the stiffness matrix with respect to the covering surface $\widetilde{M}$. The solution vector $\tilde{\mathbf{v}}$ contains the value of the co-potential $\tilde{v}$ of the curl*-part of $\widetilde{X}$ at the edge-midpoints of $\widetilde{M}$. Subtracting $\mathbf{J}\nabla\tilde{v}$ from $\widetilde{X}$ then yields an integrable field $\hat{X}$, and the potential function of $\hat{X}$ is our initial parameterization $\widetilde{\varphi}$.

What remains is to ensure the global continuity of $\widetilde{\varphi}$. Again, the procedure is analog to Chapter 2: With respect to the dual cycles of a homology basis $\tilde{\gamma}_1, \ldots, \tilde{\gamma}_{2\tilde{g}}$ of $\widetilde{M}$ we measure the periods $\mathbf{b}_1, \ldots, \mathbf{b}_{2\tilde{g}}$ of the vector field $\hat{X}$. Next, we add a harmonic field to $\hat{X}$ with periods $[\mathbf{b}_i] - \mathbf{b}_i$, so that the resulting field $\bar{X}$ has integer periods $[\mathbf{b}_i]$. The integrand $\bar{\varphi}$ of $\bar{X}$ is the final parameterization function.

The next section will detail how to find a suitable homology basis on $\widetilde{M}$ and which of these basis elements are essential for QuadCover parameterization. All other steps of QuadCover are analogously to Chapter 2.

4.2 Cut Graphs on Coverings

Algorithm 1.16 gives a construction method for a homotopy basis Γ on a surface M. We could of course apply Algorithm 1.16 to $\widetilde{M}$ to retrieve a homotopy basis on its covering $\widetilde{M}$, but it is also possible to construct a cut graph $\widetilde{\Gamma}$ of $\widetilde{M}$ directly, without computing the explicit combinatorics of the covering surface $\widetilde{M}$. Our construction will also respect required symmetry properties of the homology basis.

We call a path $\tilde{\gamma}$ on $\widetilde{M}$ **symmetric** if the reversal of the path agrees with a shift by two layers (in other words, $\tilde{\gamma}$ has a constant-velocity parameterization over $[-1, 1]$ so that $\tilde{\gamma}^k(t) = \tilde{\gamma}^{k+2}(-t)$). The path $\tilde{\gamma}$ is called **asymmetric** otherwise. The differentiation between symmetric and asymmetric paths is essential here: Since all previous steps of the QuadCover algorithm respect the complex structure of vector fields on the covering (in the sense that $\widetilde{X}^0 = -\widetilde{X}^2$), also the harmonic correction field $\hat{X}$ in the last step should respect that symmetry. There are symmetric and asymmetric homology generators, and only the symmetric ones yield harmonic vector fields when constructing harmonic fields from cycles as explained in Section 1.2.5.

We will give construction rules to construct homologously independent sets of symmetric and asymmetric paths. Since the total number of paths matches the genus of $\widetilde{M}$ as given in Lemma 3.5, we conclude that the given paths constitute a homology basis. However, we will only use the symmetric paths for QuadCover.

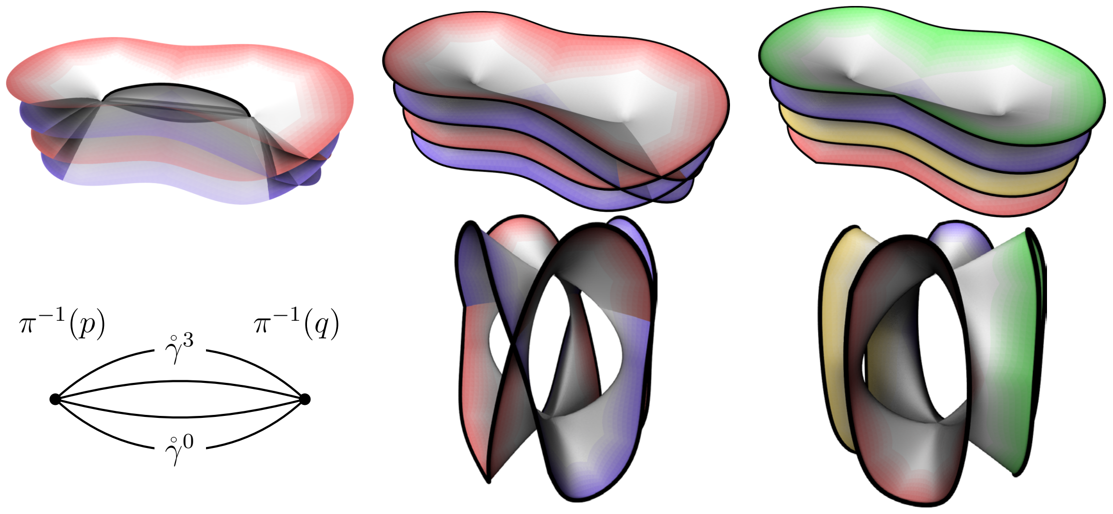


Figure 4.1: If $\gamma \in \Gamma$ connects two vertices $p, q \in B$ then $\pi^{-1}(\gamma)$ consists of four paths connecting $\pi^{-1}(p)$ and $\pi^{-1}(q)$ in $\widetilde{M}$. *Left*: A cross section through two branch points of the covering surface (top) reveals the topology of $\pi^{-1}(\gamma)$ (bottom). *Center and right*: The cases $\text{ind}_p = \text{ind}_q = \pm\frac{1}{4}$ and $\text{ind}_p = -\text{ind}_q = \pm\frac{1}{4}$. They show the corresponding covering surfaces on top and the (topologically) same surfaces in an untangled state below, so that their topology is revealed. Each boundary component was given a distinct color. See Figure 4.2 for the remaining cases.

Construction

We first construct a system of cut paths $\Gamma = \{\gamma_1, \dots, \gamma_n\}$ on M with a given set B of base vertices consisting of all branch points and boundary vertices of M (using our generalization of Algorithm 1.16). By construction, Γ will cut the surface M to a disk, and the total length of the paths in Γ is minimal among all sets of cut paths based at B. Furthermore, the construction assures that no path of Γ will contain any vertices of B aside from its end points.

Let $\pi : \widetilde{M} \to M$ be the covering map of $\widetilde{M}$, according to Definition 3.4. As a covering map, π possesses the following *lifting property* [Mas64]: Every path in M has its unique lift in $\widetilde{M}$ if a start point on $\widetilde{M}$ is given. Formally, if $\gamma : [0,1] \to M$ is a path in M and $\tilde{p} \in \pi^{-1}(\gamma(0))$ a point in $\widetilde{M}$ "lying over" $\gamma(0)$, then there exists a unique path $\tilde{\gamma}$ in $\widetilde{M}$ with starting point $\tilde{p}$ lying over γ (*i. e.*, $\tilde{\gamma}(0) = \tilde{p}$ and $\pi \circ \tilde{\gamma} = \gamma$). The path $\tilde{\gamma}$ is called the **lift** of γ.

For a path $\gamma_i \in \Gamma$, consider the four lifts $\tilde{\gamma}_i^0, \dots, \tilde{\gamma}_i^3$ of γ_i in $\widetilde{M}$. Figures 4.1 and 4.2 list all cases that can arise. In each of the configurations, the paths $\tilde{l}_i^k := \tilde{\gamma}_i^k \cup \tilde{\gamma}_i^{k+2}$ constitute either a loop or a path with both endpoints on the boundary. We claim that for a fixed k, the loops

$$\tilde{l}^0_1,\ \tilde{l}^1_1,\ \ldots, \tilde{l}^0_{n\text{-}2},\ \tilde{l}^1_{n\text{-}2},\ \tilde{l}^0_{n\text{-}1},\ \tilde{l}^1_{n\text{-}1}$$

are all non-homologous.

Note that we used only $n-1$ paths of Γ: If we were to use all n paths of the cut graph Γ, then the concatenation of $\tilde{l}^0_1, \ldots, \tilde{l}^1_n$ would bound the layers k and $k+2$ of $\pi^{-1}(M \setminus \Gamma)$, each of which is a topological disk, so the concatenation is separating. The concatenation is therefore a non-trivial linear combination of the identity element and thus $\tilde{l}^0_1, \ldots, \tilde{l}^1_n$ is not homologously independent.

Let again b_{even} and b_{odd} be the numbers of branch points with even and odd index, respectively. The maximal number of homologously independent loops on $\widetilde{M}$ equals twice the genus $\tilde{\mathsf{g}}$ of $\widetilde{M}$, which is

$$2\tilde{\mathsf{g}} \ = \ 8\mathsf{g} + \mathsf{b}_{\text{even}} + 3\mathsf{b}_{\text{odd}} - 6$$

by Lemma 3.5. Note that if b_{odd} happens to be zero, we have to add 2 (if $\mathsf{b}_{\text{even}} > 0$) or 6 (if $\mathsf{b}_{\text{even}} = 0$) to the above formula for $2\tilde{\mathsf{g}}$, but to not get lost in distinction of cases, let us always assume the existence of branch points of odd index, *i. e.*, $\mathsf{b}_{\text{odd}} > 0$.

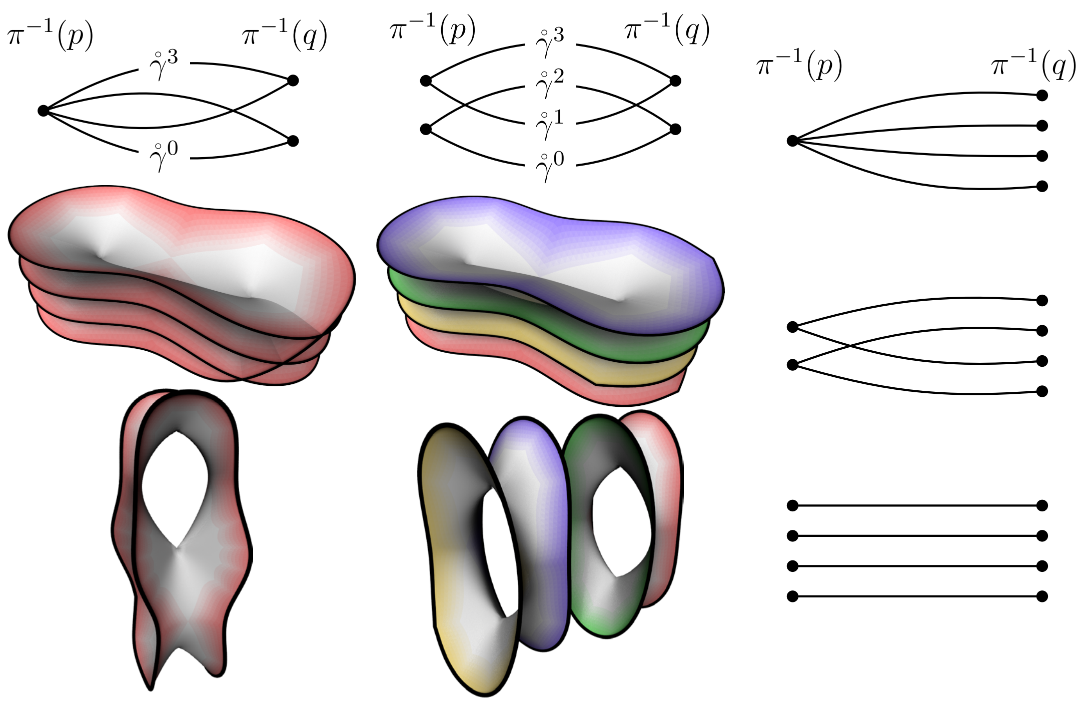
Our above construction gives rise to

$$\tilde{\mathsf{h}}_{\text{sym}} := 4\mathsf{g} + 2(\mathsf{b}_{\text{even}} + \mathsf{b}_{\text{odd}}) - 4 \tag{4.1}$$

symmetric loops so $4\mathsf{g} + \mathsf{b}_{\text{odd}} - 2$ paths still remain. These remaining paths are all asymmetric, and thus are not of interest for us. However, we briefly describe their construction for completeness.

$4\mathsf{g}$ of these asymmetric paths are to be constructed from the loops in Γ that arise from the handles of M. The remaining paths correspond to branch points with odd layer shift. In the preimage of $\pi^{-1}(\gamma)$ of a path between two such branch points, three independent loops can be constructed, compare Figure 4.1. Of those three loops, we already used two symmetric ones, $\tilde{\gamma}^k \cup \tilde{\gamma}^{k+2}$ and $\tilde{\gamma}^{k+1} \cup \tilde{\gamma}^{k+3}$. There is one more independent asymmetric loop in $\pi^{-1}(\gamma)$, for example $\tilde{\gamma}^k \cup \tilde{\gamma}^{k+1}$.

The construction of a loop $\tilde{\gamma}^k \cup \tilde{\gamma}^{k+1}$ fails if one of the branch points has an even layer shift (cases (a) and (b) in Figure 4.2), but these cases can be avoided altogether when using a different path layout, as we used in the original QuadCover paper. In the original QuadCover path layout, all paths started in a single "base" branch point of odd layer shift. So for each odd singularity except the base point we get an asymmetric path. Of those $\mathsf{b}_{\text{odd}} - 1$ paths to branch points of odd index, there is one which linearly depends on the others, resulting in the predicted total of $4\mathsf{g} + \mathsf{b}_{\text{odd}} - 2$ independent asymmetric paths.



(a) $\mathrm{ind}_p = \pm\frac{1}{4}$, $\mathrm{ind}_q = \frac{1}{2}$ (b) $\mathrm{ind}_p = \mathrm{ind}_q = \frac{1}{2}$ (c) The three cases for $q \in \partial M$: $\mathrm{ind}_p = \pm\frac{1}{4}$, $\mathrm{ind}_p = \frac{1}{2}$, and $p \in \partial M$

Figure 4.2: See Figure 4.1.

4.2.1 Global Continuity

Let us get back to the symmetric cycles, which we need for the construction of symmetric harmonic fields. We now repeat the same steps of Section 1.2.6, to compute harmonic fields, but we compute them on the covering. First, we construct harmonic fields on $\widetilde{M}$ by generating the vector fields $\widetilde{\Sigma}_i$ along the left side of a homology generator $\tilde{\gamma}$. We then set $\widetilde{H}_i := \widetilde{\Sigma}_i - (\widetilde{\Sigma}_i)_{\widetilde{\mathcal{P}}}$ is a field with a period of 1 with respect to $\tilde{\gamma}^*$. To generate the harmonic correction field $\widetilde{H}_{\mathbf{d}}$ for the global continuity as we did in Section 2.4.2 for the unbranched case, we compute $\widetilde{H}_{\mathbf{d}} = \widetilde{\Sigma}_{\mathbf{d}} - (\widetilde{\Sigma}_{\mathbf{d}})_{\widetilde{\mathcal{P}}}$, where $\widetilde{\Sigma}_{\mathbf{d}} = \sum_i \mathbf{d}_i \widetilde{\Sigma}_i$ is chosen so that $\hat{X} + \widetilde{\Sigma}_{\mathbf{d}}$ has integer periods. Figure 4.3 shows a harmonic vector field constructed on the covering from one of the symmetric cycles.

To put the method in relation to the cycles we just constructed, we let $\widetilde{\Sigma}_i^0$ and $\widetilde{\Sigma}_i^1$ be the path that correspond to $\tilde{\gamma}_i^0 \cup \tilde{\gamma}_i^2$ and $\tilde{\gamma}_i^1 \cup \tilde{\gamma}_i^3$, respectively, via the construction of (1.13). Note that $\widetilde{\Sigma}_i^0$ is not symmetric, because the support of $\widetilde{\Sigma}_i^0$ is different the layers 0 and 2, due to a different 'left side' under a reversal of $\tilde{\gamma}_i$. Its harmonic part $\widetilde{H}_i^0 := \widetilde{\Sigma}_i^0 - (\widetilde{\Sigma}_i^0)_{\widetilde{\mathcal{P}}}$ *is* harmonic, though: To show this claim we let $\widetilde{\Sigma}_{-i}^0$ be the vector

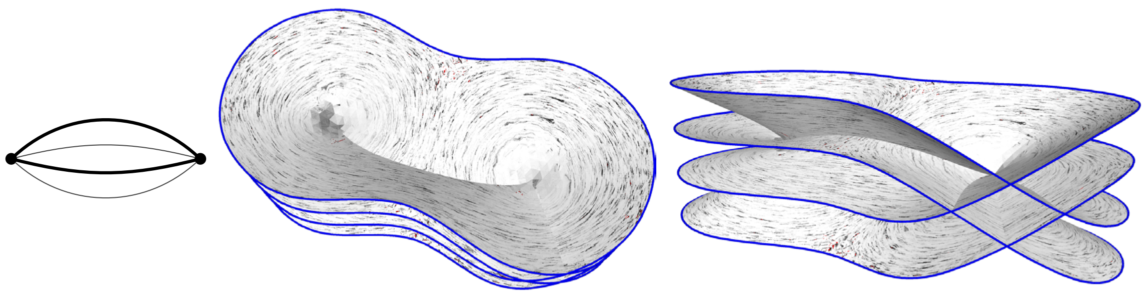


Figure 4.3: The harmonic basis field constructed from the cycle through layers 1 and 3, shown from two perspectives. One can see that the field flows perpendicular to the paths in layers 1 and 3, but parallel to the paths in layers 0 and 2. (The paths themselves is not displayed here, but shown in Figure 4.1, top left.)

field with respect to the cycle $\tilde{\gamma}_i^0 \cup \tilde{\gamma}_i^2$ with reversed orientation. Then $\widetilde{\Sigma}_i' := \frac{1}{2}(\widetilde{\Sigma}_i^0 - \widetilde{\Sigma}_{-i}^0)$ has the same periods as $\widetilde{\Sigma}_i^0$, so they have the same harmonic part. Due to the symmetry of $\widetilde{\Sigma}_i'$, the fields $(\Sigma_i')_{\widetilde{\mathcal{P}}}$ and thus, $\widetilde{H}_i = \widetilde{H}_i' := \widetilde{\Sigma}_i' - (\widetilde{\Sigma}_i')_{\widetilde{\mathcal{P}}}$ must be symmetric too, which verifies the claim. The same reasoning applies of course to $\widetilde{\Sigma}_i^1$ as well.

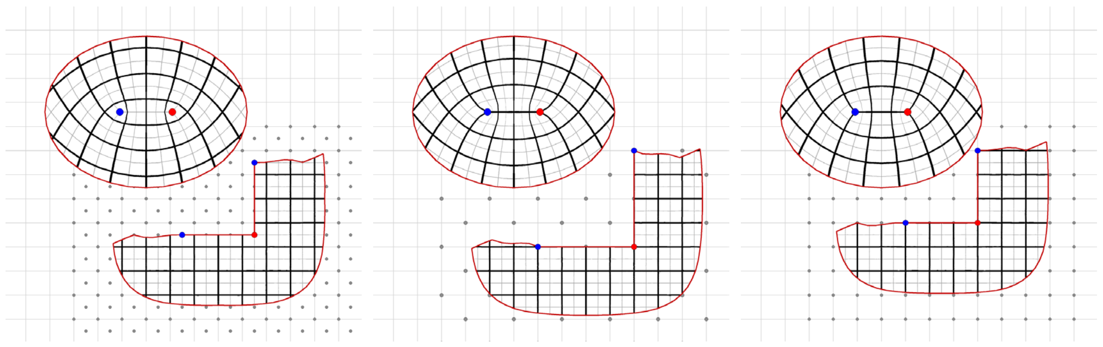
Again, complex notation can be convenient to denote the period vectors of the vector fields on the covering. Hence, we use one complex number $\mathbf{b}_i$ to denote the periods of our vector field $\widetilde{X}$, with respect to the paths $\tilde{\gamma}_i^0 \cup \tilde{\gamma}_i^2$ and $\tilde{\gamma}_i^1 \cup \tilde{\gamma}_i^3$. Finally, after correcting the vector field to have integer periods on the covering, we can integrate the field to yield our final parameterization $\bar{\varphi} = \varphi^0 + \boldsymbol{i}\varphi^1 : M \to \mathbb{C}$. This completes the essential parts of the QuadCover algorithm.

4.2.2 Pure Quadrilateral Meshes

Rounding the periods $\mathbf{b}_i$ to complex integers (also called Gaussian integers, denoted as $\mathbb{Z}[\boldsymbol{i}] = \{a + b\boldsymbol{i} \mid a, b \in \mathbb{Z}\}$) ensures that all parameter lines are visually continuous. However, this does not guarantee all-quadrilateral meshes, because parameter lines could form, for example, a triangle or pentagon around branch points whose parameter positions happen to lie at half-integers, see Figure 4.4, left.

During the rounding procedure, branch points are forced to positions in the parameter space which are fix points with respect to a rotation of the integer lattice. Branch points of odd index are forced to either grid points or grid midpoints, as those point remain grid (mid)points under rotations of $\pi/2$ of the parameter grid. Branch points of even index may additionally lie on edge midpoints of the parameter grid, because those point hold the rotational invariance with respect to half rotations but not with respect to quarter rotations.

The algebraic explanation is the following: The value of the parameter function of a branch point v is determined by the periods of the paths $\tilde{\gamma}_1^0, \tilde{\gamma}_1^1, \ldots, \tilde{\gamma}_n^0, \tilde{\gamma}_n^1$ joining v.



(a) $\mathbf{b} \in (\mathbb{Z} \times \mathbb{Z})^{\tilde{\mathfrak{h}}_{\mathrm{sym}}}$ (b) $\mathbf{b} \in 2(\mathbb{Z} \times \mathbb{Z})^{\tilde{\mathfrak{h}}_{\mathrm{sym}}}$ (c) $\mathbf{b}$ is in the lattice generated by $\{(1,1),(-1,1)\}^{\tilde{\mathfrak{h}}_{\mathrm{sym}}}$

Figure 4.4: The figures show a parameterized ellipse and the surface unfolded to the parameter space, using different rounding methods for the period vector **b**. The shown lattice points mark the possible positions of branch point with odd index.

If their periods are set to $\tilde{x}_1^0, \tilde{x}_1^1, \ldots, \tilde{x}_n^0, \tilde{x}_n^1$, then the position of v in the complex parameter space is given by

$$\begin{aligned} &\sum_{j=1}^{n} \tfrac{1}{2}\big((x_j^0 \pm x_j^1) \mp \boldsymbol{i}\,(x_j^0 \pm x_j^1)\big) && \text{if } \mathrm{ind}_v \text{ is odd} \\ &\sum_{j=1}^{n} \tfrac{1}{2}\big(x_j^0 + \boldsymbol{i}\,x_j^1\big) && \text{if } \mathrm{ind}_v \text{ is even} \end{aligned}$$

Thus, if all periods are integers and ind_v is odd, then the real and imaginary coordinates are not necessarily the same, but they are always have the same decimal fraction (which is either 0 or 0.5). The coordinates of branch points of even index can take any values in $\frac{1}{2}\mathbb{Z}[\boldsymbol{i}]$.

Now the easiest solution to guarantee integer coordinates is to round twice as coarse: Instead of rounding the coordinates of the period vector **b** to values in $\mathbb{Z}[\boldsymbol{i}]$, we round them to values in $2\mathbb{Z}[\boldsymbol{i}]$, and so all branch points will have integer coordinates, see Figure 4.4, center.

To round the vector **b** more accurately (in terms of smaller rounding errors) we observe the following: If we round **b** to the regular Gaussian integer lattice $\mathbb{Z}[\boldsymbol{i}]^{\tilde{\mathfrak{h}}_{\mathrm{sym}}}$, then all branch points of odd index will lie on a rotated and downscaled integer lattice generated by the lattice vectors $\{(0.5, -0, 5), (0.5, 0.5)\}$. Thus, to reverse the effect, we round **b** to entries in a rotated and upscaled lattice generated by $\{(1,1),(-1,1)\}^{\tilde{\mathfrak{h}}_{\mathrm{sym}}}$, so all odd branch points end up on complex integer points, see Figure 4.4, right. The lattice generated by $\{(1,1),(-1,1)\}$ has a higher density of grid points than $2\mathbb{Z}[\boldsymbol{i}]$, so the expected rounding artifacts will be smaller.

Branch points of even index (which usually make up a tiny fraction of branch points) need extra attention since their positions may still end up on a grid midpoint when rounding $\mathbf{b}$ to $\{(1,1),(-1,1)\}^{\tilde{\mathfrak{h}}_{\mathrm{sym}}}$. Properly adjusting the period of one path per branch point of even index can properly move the respective branch points to integer coordinates if this situation is observed.

Chapter 5

Frame Field Generation

5.1 Principal Curvature Fields

The Hodge decomposition in the QuadCover parameterization relies on the existence of a frame field on the surface. For each frame field QuadCover delivers a surface parameterization but of course, the quality of the result depends on the suitability of the input frame field. By having a method to find the best-fitting parameterization for a given input frame field, QuadCover reduces the parameterization problem to finding a suitable input frame field. The latter problem is still far from trivial, given the numerous conflicting objectives of good parameterizations listed in Section 2.1.

In our QuadCover paper [KNP07], we chose to use principal curvature directions as the input directions for QuadCover, as they naturally support the *regularity* and *alignment quality* requirement, by being naturally perpendicular, normalized to unit length, and aligned to provide good surface approximation. Furthermore, lines of principal curvature look aesthetic, as they support the human perception of curved shapes, which is why they are often used in arts, see Hertzmann and Zorin [HZ00].

5.1.1 Computing Principal Curvature Directions

Unlike in the smooth case, the PL setting does not provide a notion of *the* principal curvature directions of a surface. There is a wealth of methods to estimate the principal curvature directions, though, some of which converge to the smooth limit. In our QuadCover implementation we used two methods to compute one principal curvature frame per triangle:

The simpler method averages the three vertex-based shape operators of Hildebrandt and Polthier [HP04] in each triangle to get a face-based tensor. The unit-length eigendirections of this tensor then constitute the principal curvature frame field. The

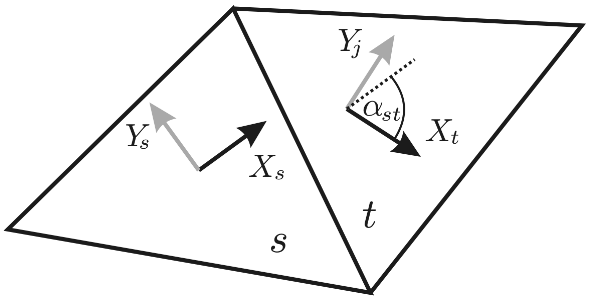


Figure 5.1: Computing the matching: With an angle of $\alpha_{st} := \angle(X_s, X_t)$ of about 70 degrees we get a matching of 1 by Formula 5.1. This corresponds to the fact that the frame (X_t, Y_t) is most parallel to $R^{\mathbf{1}}(X_s, Y_s)$.

second algorithm is based on *Restricted Delaunay Triangulations and Normal Cycle* by David Cohen-Steiner and Jean-Marie Morvan [CSM03]. Their method computes the shape operator at a point by averaging the edge-based tensors within a disk of a given radius. This is more complex to implement, but results are much better on noisy surfaces.

Both methods yield a perpendicular, normalized cross field which needs to be endowed with a matching. Initially, we choose each matching m_{st} via

$$m_{st} := \text{round}\big(2\angle(X_s, X_t)/\pi\big) \tag{5.1}$$

according to Section 3.1.2, so that the resulting curvatures $\theta_{st} = \angle(X_s, X_t) + \frac{\pi}{2} m_{st}$ have values between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Remember that $\angle(X_s, X_t) \in [-\pi, \pi]$ is measured intrinsically.

5.1.2 Regions of Stable Curvature Directions

Principal curvature directions turned out to be a powerful choice for guiding frame fields, but their usefulness is limited to regions in which the principal curvature directions can be computed stably. Two obstructions prevent the robust computation of principal curvature directions: noise and umbilic regions, *i. e.*, flat regions or spherical caps, in which the directions of minimal and maximal curvature are undefined.

To avoid that directions in stable regions get spoiled by those in unstable ones, we use a triangle-based stability measure of curvature frames [KP10]. Our method is similar to that of Bommes *et al.*, with the difference that we use a continuous scale instead of having only trusted and non-trusted triangles. Our stability weights are defined by:

$$\omega_t^{\text{stable}}(F) = \left|\kappa_t^{\max} - \kappa_t^{\min}\right| \cdot \exp\!\big(-\tfrac{1}{2}(|\theta_{e_1}| + |\theta_{e_2}| + |\theta_{e_3}|)\big)\,. \tag{5.2}$$

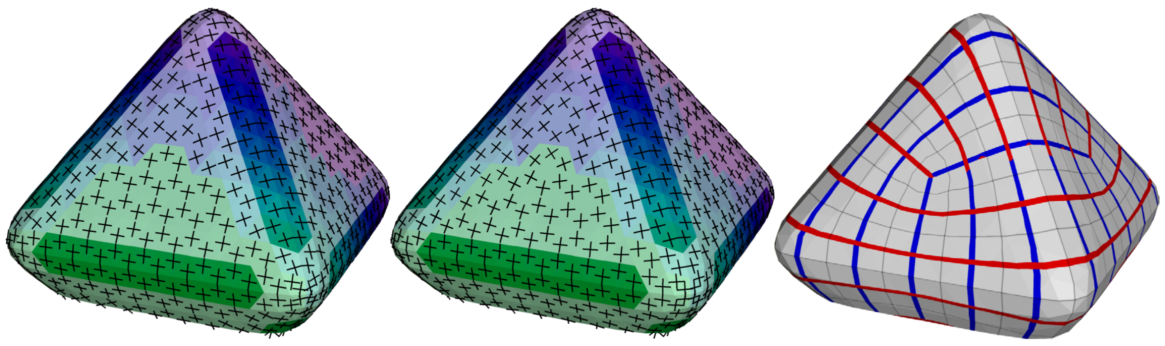


Figure 5.2: *Left:* The frames from the dark stable regions were extended to the light regions. The colors indicate from which dark elements the frames were extended. *Center:* Smoothing is applied to the frames. *Right:* The resulting parameterization

The values $\kappa_{\max}$ and $\kappa_{\min}$ are the principal curvature values, so the left factor measures how far the shape is from being umbilic. The edges e_1, e_2, and e_3 are the incident edges of t. High values of θ at the edges reliably indicate surface noise, and so affected triangles will have a low stability value. The second factor is maximal if the principal curvature directions are parallel to those in the neighboring triangles.

With a characterization of stable curvature directions, we can stick with a user-defined percentage of the most stable directions and ignore the other ones. Keeping 10–30% of the most stable curvature frames provides good results in our experience, but this value depends on the particular surface model. To fill the rest of the triangles with direction frames, we extend the frames from the stable regions in a breadth first manner, parallel transporting the frames across the edges.

The parallel transported frames tend to be much less noisy (in the sense of lower curvature across the edges) than pure principal curvature frames. As a result, the matching determined via (5.1) vanishes more often and thus causes less singularities. On the other hand, the parallel transport leads to “seams" when direction vectors that are propagated form different regions meet together, as seen in Figure 5.2. That is one of the reasons why we smooth the direction field by rotating each frame in its tangent plane, as explained next.

5.2 Smoothing Frame Fields

To avoid peaks of frame field curvature θ, we rotate the frames in the individual tangent planes to distribute the rotational difference over larger areas. We use the smoothing energy

$$E_{\mathrm{S}} := \sum_{e \in \mathsf{E}} \omega_e \, \theta_e^2, \qquad \text{with} \quad \omega_e = \frac{|\mathbf{e}|^2}{A_{\mathbf{e}}},$$

of the Mixed Integer Quadrangulation [BZK09] to define the total smoothness of a frame field, but we add the factor ω_e to compensate the effects of varying triangle sizes and shapes. The factor $A_{\mathbf{e}} = \frac{1}{3}(A_{\mathbf{s}} + A_{\mathbf{t}})$ is the share of the surface area of an edge e between s and t. Since

$$\omega_e\, \theta_e^2 \;=\; 4\, A_{\mathbf{e}}\, (\theta_e/\text{height}_{\text{star }e})^2,$$

E_{S} can be understood as curvature per length unit across e, integrated over the edge star areas. We use this energy for two reasons. First, the *regularity* requirement asks for straight parameter lines, so we minimize the curvature of the guidance field in a least squares sense.

Second, if the curl of the direction field is low, then QuadCover's curl removal will alter the input field much less. As shown by the following calculation, the curvature of an orthogonal, direction field is a good approximation of the frame field curl. Consider the squared length of the curl vector of the lift $\widetilde{X}$ the input field,

$$\begin{aligned}
\|\mathbf{curl}^*\widetilde{X}\|^2 \;&=\; \sum_{e\in\mathsf{E}}\sum_{l=0}^{3}\Big(\text{curl}^*\widetilde{X}(e^l)\Big)^2 \\
&\overset{(1.6)}{=}\; \sum_{e\in\mathsf{E}}\sum_{l=0,2}\Big(\text{curl}^*\widetilde{X}(e^l)\Big)^2 + \Big(\text{div}^*\widetilde{X}(e^l)\Big)^2 \\
&=\; 2\sum_{e\in\mathsf{E}}\Big(\text{curl}^*\widetilde{X}(e^0)\Big)^2 + \Big(\text{div}^*\widetilde{X}(e^0)\Big)^2
\end{aligned}$$

Let θ_e be the curvature of $\widetilde{X}$ at e and let $\alpha = \angle(X_s, \mathbf{e})$ as depicted in Figure 5.3.

$$\begin{aligned}
\|\mathbf{curl}^*\widetilde{X}\|^2 \;&=\; 2\sum_{e\in\mathsf{E}}\big(\cos(\alpha) - \cos(\alpha+\theta_e)\big)^2\|\mathbf{e}\| + \big(\sin(\alpha) - \sin(\alpha+\theta_e)\big)^2\|\mathbf{e}\| \\
&=\; 2\sum_{e\in\mathsf{E}}\left(2 - 2\cos(\theta_e)\right)\|\mathbf{e}\| \\
&=\; 2\sum_{e\in\mathsf{E}}\theta_e^2\,\|\mathbf{e}\| + O(\theta_e^4)\|\mathbf{e}\|.
\end{aligned}$$

Thus, up to the weighting factor, the smoothing energy is a third order polynomial approximation to $\|\mathbf{curl}^*\widetilde{X}\|^2$, if $\text{curl}^*\widetilde{X}$ is seen as a vector in $\mathbb{R}^{\tilde{\mathsf{e}}}$. Interestingly, $\|\mathbf{curl}^*\widetilde{X}\|$ depends only on the curvature θ_e and not on the angles between frames and edges.

5.2.1 Minimizing E_{S}

To find the smoothest field with respect to E_{S}, we start with the frame field that we gained from extending the principal curvature directions from the trust regions. For each triangle t of M we let α_t be our free variable that describes by how much we

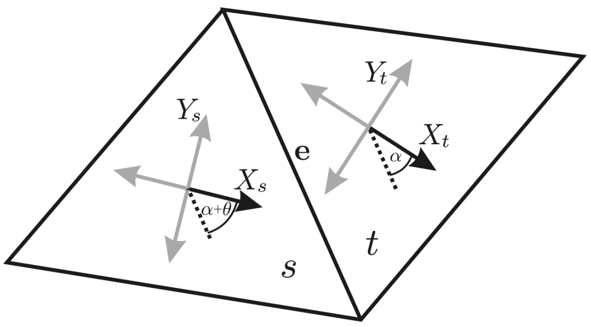


Figure 5.3: Angles α and θ.

rotate the frame in triangle t. We then minimize the following smoothing energy:

$$E(\alpha) = \lambda E_{\mathrm{S}}(\alpha) + (1-\lambda) E_{\mathrm{A}}(\alpha). \tag{5.3}$$

The smoothing term

$$E_{\mathrm{S}}(\alpha) = \sum_{(s,t)\in \mathsf{E}} \omega_{st}(\theta_{st} - \alpha_s + \alpha_t)^2$$

is as above, but takes the rotation of the input field by angles α_t into account. The additional alignment term

$$E_{\mathrm{A}}(\alpha) = \sum_{\substack{\text{trusted} \\ \text{triangles } t}} \omega_t^{\text{stable}} \alpha_t^2 \, A_{\mathbf{t}}.$$

makes sure that the frame field does not deviate too much from its original directions in the trusted regions. Via the parameter $\lambda \in [0,1]$ the amount of smoothing can be controlled from the complete ignorance of the original curvature directions to the other extreme of fixed directions in trust regions.

Since the energy is quadratic, the minimizer can be found by solving the sparse linear system $\mathbf{A}\alpha = \mathbf{b}$, with a sparse matrix $\mathbf{A} \in \mathbb{R}^{\mathsf{f}\times\mathsf{f}}$,

$$\mathbf{A}_{st} = \begin{cases} \lambda \sum_{s\sim t} \omega_{st} \; + \; (1-\lambda) w_t^{\text{stable}} A_{\mathbf{t}} & \text{if } s = t \\ -\lambda \omega_{st} & \text{if } s \sim t \end{cases} \tag{5.4}$$

and a right-hand side vector

$$\mathbf{b}_t = \lambda \sum_{s\sim t} \omega_{st} \cdot \theta_{st}.$$

5.2.2 Branch Point Relocation

Since the position of branch points depends only on the matching, we can influence the branch point positions by changing the matching on individual edges. However, changing the matching without rotating the frames will violate the property that vectors are matched in the straightest way. Re-computing the minimizer of $E(\alpha)$ will adapt the frame field to the new matching.

If a matching m_{st} between triangles s and t is changed by a value k to $m_{st} + k$, we must only update the right-hand side of the linear system:

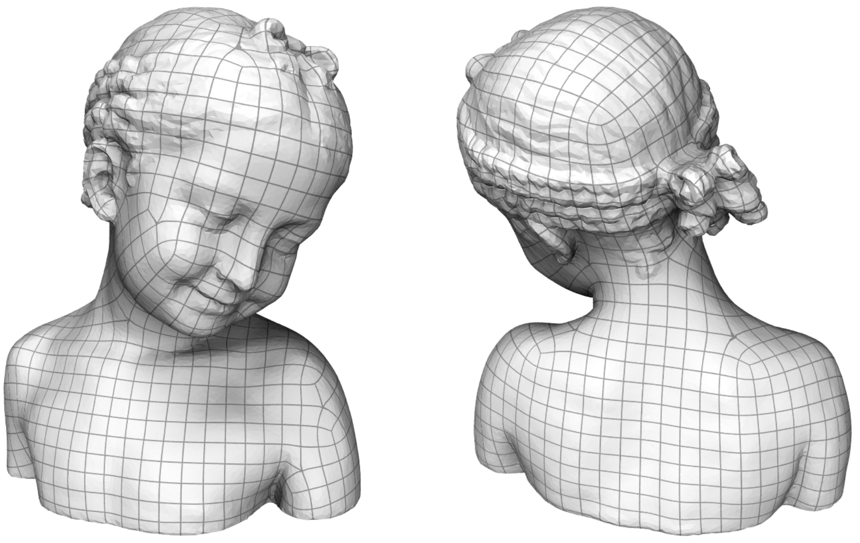
$$\mathbf{b}_s \leftarrow \mathbf{b}_s + \tfrac{\pi}{2} k \lambda \omega_{st} \qquad \mathbf{b}_t \leftarrow \mathbf{b}_t - \tfrac{\pi}{2} k \lambda \omega_{st}.$$

The matrix $\mathbf{A}$ in Equation 5.4 does not depend on the matchings. Therefore, we can store a Cholesky factorization of the matrix, making it extremely fast to solve the system for multiple right hand sides. Once a matching is changed from its initial choice via (5.1), it is not guaranteed that (5.1) holds again after smoothing is applied (particularly, if the smoothing parameter λ is zero), so we compute it once in the beginning, and update it as needed.

Moving a branch point from vertex p to vertex q is accomplished by finding an edge-based path from p to q and adding $4\,\mathrm{ind}_m(p)$ to the matching of all the edges on the path (if the edge orientation is from right to left when looking from p to q).

Chapter 6

Minimizing Distortion



6.1 Branch Point Placement

Because the branch point placement substantially influences the characteristics of the frame field, many methods have been proposed for branch point placement.

For example, the aforementioned method of Ben-Chen *et al.* [BCGB08] is looking for the best branch point positions for quasi-conformal parameterizations. The length distortion (the so-called *conformal factor*) is mainly determined by the branch point positions. Ben-Chen et al. placed the branch points at the positions of the worst length distortion, and could so keep the length distortion very low. They also inspired Springborn et al. [SSP08] to further improve the branch point positioning by iterative relocation, as well as Myles and Zorin [MZ12], who also aimed for a uniformly distributed scale factor, but iteratively fixed vertices that may *not* become branch points, so that the branch point candidates gradually concentrated at a small set of vertices.

More closely related to our parameterization is the Mixed Integer Quadrangulation (MIQ) by Bommes, Zimmer, and Kobbelt [BZK09], who combined the problem of choosing a suitable matching (and thus, the branch point positions) and smoothing the field in a mixed integer problem. They start with a non-integer matching, and greedily rounded the matching until they are all integral.

The very recent work of Myles and Zorin [MZ13] aims to combine the advantages of their previous method [MZ12] and MIQ in a unified framework. They find a closed form representation of smoothness-maximizing one forms, that allows to respect directional constraints exactly. This approach is very elegant, as this eliminates the need the need of a smoothing step to minimize $\|\theta\|^2$ that we use in 5.2. Because the smoothing respects only hard constraints, it is suited well for sharp-edged surfaces, rather than for organic shapes, on which appropriate hard constraints difficult to define. Their framework allows to directly measure the distortion inherent to a parameterization similar to the size of the curl part that we use for optimization in [KP10, Nie12]. The size of the curl part will also play a major role later in this Chapter

In QuadCover, we determine the branch points via the matching of the (parallel transported) direction field. This method is very simple, and it respects the given input directions for the branch point placement (of the above methods, only MIQ has this property). We will use this simple branch point placement as a starting point, but combine it with combination of powerful heuristics that incrementally improve their number and positions. I consider the efficient and effective branch point optimization is one of the main contributions of this thesis, next to the QuadCover method itself. Together, this will lead to much lower distortion than that of existing methods.

6.1.1 A Distortion Measure

In Chapter 5, we discussed how to produce smooth, feature-aligned and noise-free guiding frame fields $X \in \mathfrak{F}$. If the guiding field should happen to be additionally curl free, we could—at least locally—generate a perfect parameterization in the sense that the isolines of the parameterization would exactly keep the guiding field's directions, because the curl removal part of the Hodge decomposition would not affect the field at all. Moreover, if we started with a constant-length cross field, then the parameterization map would be an isometry, so *all* of the requirements of Chapter 2 could be met in this ideal case.

Of course, this ideal parameterization is not achievable in general. The size of the curl part of X, $\|X_{\mathcal{C}}\|$, measures how far X is away from being curl free, *i. e.*, from being ideal. At the same time, $\|X_{\mathcal{C}}\|$ is the amount of modification that has to be made to make X locally integrable, so we will speak of $\|X_{\mathcal{C}}\|^2$ as the amount of distortion of the first QuadCover step, which we denote by

$$E_D(X) := \frac{1}{2}\|X_\mathcal{C}\|^2.$$

We have used the term E_D before to denote the Dirichlet energy of a function v, but since $E_D(X)$ *equals* the Dirichlet energy of the co-potential of $X_\mathcal{C}$, this is consistent.

In particular, $\|X_\mathcal{C}\|^2$ includes the *as-rigid-as-possible* energy (ARAP energy) used in Liu *et al.* [LZX+08]. The ARAP energy measures the difference between the parameterization $\nabla\varphi$ and the best-fitting unit-length cross in each triangle,

$$E_{\text{ARAP}} = \frac{1}{2}\sum_t \min_{R\in SO(2)} \|\nabla\varphi_t - R\|^2 A_\mathbf{t},$$

and so is always smaller than E_D, which does not allow the best-fit rotation in each triangle individually.

The curl of a cross field X also coincides with the divergence of the field, up to discretization: If the frame field is perpendicular, we have $X = \mathbf{J}X$ up to the numbering of layers. Therefore, if $v \in S_h^*(\widetilde{M})$ is the curl* co-potential, *i. e.*, v minimizes $\int_{\tilde{M}} \|X - \mathbf{J}\nabla v\|^2 \,\mathrm{d}A$, then v is also a minimizer of $\int_{\tilde{M}} \|X - \nabla v\|^2 \,\mathrm{d}A$ among the non-conforming basis functions, which makes v a div* potential of X. Thus, the minimizer of E_D also reduces the (non-conforming) divergence. As a consequence, a unit-length cross field that minimizes E_D, maximizes its harmonic part at the same time (again, under the assumption that the discretization div$^*X$ does not differ significantly from div$X$).

After the curl removal in QuadCover's first step, the next step is to assure *global continuity* by adding a harmonic frame field $H_\mathbf{d}$. This will necessarily introduce additional distortion, which we denote by

$$E_H = \frac{1}{2}\|H_\mathbf{d}\|^2.$$

Together, the two introduced errors describe the discrepancy between and the ideal guiding field and the final parameterization φ, *i. e.*, $E_D + E_H = \frac{1}{2}\int_{\tilde{M}} \|X - \nabla\varphi\|^2 \,\mathrm{d}A$.

In the following sections, we will describe how each of the two components of the distortion can be kept small, starting with the curl part, E_D.

6.2 Curl Minimization

For given trusted curvature directions and fixed matching, the smoothing method of Section 5.2 smoothes a frame field, such that the curl is approximately minimized.

Since the curvature directions are fixed, it is the matching (and the resulting branch point layout) which primarily determines the resulting parameterization.

It is taken for granted in the parameterization literature that finding an optimal matching is NP-hard. Since any greedy algorithm can only find sub-optimal solutions, it is natural to look for better ones by iteratively trying to improve the initial solution.

Our contribution to reduce E_D is two-fold: First, we show how to evaluate the distortion E_D of alternate solutions quickly (below), and second, we propose an efficient heuristic to steer the search for better solutions (Section 6.2.2).

We have already used the size of $\|X_\mathcal{C}\|$ as a useful distortion measure for curl reduction in [KP10, Nie12], but the lack of a fast recomputation and a reasonable evaluation order limited its usefulness to very small models. The key idea of the speedup are updates of the Cholesky decomposition of the stiffness matrix. But to get the updates up and running, two problems had to be solved: First, to avoid the dependence on elaborate data structures as used in CHOLMOD [CDHR08], the sparsity pattern of the matrix must be kept unchanged. Second, rank-k updates of Cholesky factorizations must be carefully split into k rank-1 updates without causing the update process to fail. Both problems are addressed below.

6.2.1 Computing the Curl

We start from an input field X that is computed as the as the minimizer of $E(\alpha)$ as in Section 5.2. The easiest way to determine the L^2-norm of $X_\mathcal{C}$ is to compute it via the Dirichlet energy of the co-potential $\tilde{v} \in S_h^*(\widetilde{M})$ (which we interpret as a complex vector $\mathbf{v} \in \mathbb{C}^\mathsf{v}$ via (3.6)):

$$\left\|X_\mathcal{C}\right\|^2 = 2E_D(\mathbf{v}) = \mathbf{v}^\mathsf{H}\,\overline{\mathfrak{L}}^*\,\mathbf{v},$$

where $\mathbf{v}$ satisfies

$$\overline{\mathfrak{L}}^*\,\mathbf{v} = -\mathbf{curl}^* X. \tag{6.1}$$

In fact, it is not necessary to solve for $\mathbf{v}$ completely: When using the Cholesky decomposition $\overline{\mathfrak{L}}^* = \mathbf{L}^\mathsf{H}\mathbf{L}$, the two systems

$$\begin{array}{rcll} \mathbf{L}\,\mathbf{x} & = & -\mathbf{curl}^* X & \quad\text{and} \\ \mathbf{L}^\mathsf{H}\tilde{\mathbf{v}} & = & \mathbf{x} & \end{array}$$

are solved successively in order to find $\mathbf{v}$. However, since

$$\left\|X_\mathcal{C}\right\|^2 = \mathbf{v}^\mathsf{H}\,\overline{\mathfrak{L}}^*\,\mathbf{v} = \mathbf{v}^\mathsf{H}\mathbf{L}^\mathsf{H}\,\mathbf{L}\mathbf{v} = \mathbf{x}^\mathsf{H}\mathbf{x},$$

we only need to solve the first of the two linear systems for $\mathbf{x}$ and so one can save the second one.

Now if we change the matching, we must first solve for the angles α which minimize $E(\alpha)$ and then compute $\|X_{\mathcal{C}}\|$ with respect to the field rotated by α. In comparison to computing $E(\alpha)$, the repeated computation of the curl component requires a matrix update whenever the matching changes, since $\overline{\mathfrak{L}}^*$ depends on the matching. Updating the Cholesky factorization instead of re-building the factorization from scratch is therefore crucial to keep the computational cost low. Luckily, the zero pattern of the stiffness matrix does not change if $\overline{\mathfrak{L}}^*$ is expressed as a complex matrix instead of using the real notation of Equation (3.7). In this case, the zero-pattern of the Cholesky factors also stays the same, keeping the sparse Cholesky update comparatively simple.

Sparse Cholesky Updates

If the matching at edge e_k between triangles t_i and t_j is changed, four entries change in the matrix: $\overline{\mathfrak{L}}^*$ has only entries in row k that correspond to edges in t_i and t_j (where we let $i < j$ without loss of generality). Since (3.5) defined the layer of e_k to equal that of t_i, the matrix entries that correspond to edge adjacencies in triangle t_i do not change. The layer difference to e_k changes only for the two edges e_l, e_m that touch e_k in t_j, so only the four matrix entries with matrix indices $(k,l), (k,m), (l,k)$, and (l,m) change at all. Let $\mathbf{D} = (\mathbf{d}_{ij})$ denote the difference between the state of $\overline{\mathfrak{L}}^*$ before and after the change of the matching. The four non-zero entries $\mathbf{d}_{kl}, \mathbf{d}_{km}, \mathbf{d}_{lk} = \bar{\mathbf{d}}_{kl}$, and $\mathbf{d}_{mk} = \bar{\mathbf{d}}_{km}$ of $\mathbf{D}$ have the form

$$\mathbf{D} = \begin{pmatrix} \cdot & \mathbf{d}_{kl} & \mathbf{d}_{km} \\ \bar{\mathbf{d}}_{kl} & \cdot & \cdot \\ \bar{\mathbf{d}}_{km} & \cdot & \cdot \end{pmatrix}$$

where $\bar{\mathbf{d}}$ denotes the complex conjugate of $\mathbf{d}$.

Cholesky updates only allow to track matrix changes of the form $\mathbf{A} \leftarrow \mathbf{A} \pm \mathbf{w}\mathbf{w}^{\mathsf{H}}$, where $\mathbf{w}$ is a complex vector of size e. As the matrix $\mathbf{w}\mathbf{w}^{\mathsf{H}}$ has rank 1, these updates are rank-one updates. In our case, the matrix $\mathbf{D}$ has higher rank, in which case we can perform the update as a series of rank-one updates. Therefore, it is necessary to find vectors $\mathbf{w}_i$, such that $\mathbf{D} = \sum \sigma_i \mathbf{w}_i \mathbf{w}_i^{\mathsf{H}}$ with $\sigma_i \in \{-1, 1\}$.

When seeking such vectors $\mathbf{w}_i$, we have to be careful to avoid a difficulty of Cholesky updates: When updating the Cholesky factors of a matrix $\mathbf{A} = \mathbf{A}_0$ to $\mathbf{A}' = \mathbf{A}_0 + \sum \sigma_i \mathbf{w}_i \mathbf{w}_i^{\mathsf{H}}$ in a series of single updates, $\mathbf{A}_{i+1} = \mathbf{A}_i + \sigma_i \mathbf{w}_i \mathbf{w}_i^{\mathsf{H}}$, it might happen that some of the matrices $\mathbf{A}_i$ are not positive semidefinite even if the first and last matrices, $\mathbf{A}$ and $\mathbf{A}'$, are themselves positive semidefinite. Since the Cholesky decomposition exists only for positive semidefinite matrices, the update chain must fail.

To avoid this pitfall, we compute the eigendecomposition $\sum \alpha_i \mathbf{v}_i \mathbf{v}_i^{\mathsf{H}}$ of the matrix $\mathbf{D}$, as proposed by Deng [Den10]. Performing the matrix updates in decreasing order of the eigenvalues will assure that the updated matrix stays positive semidefinite at each step of the update process. The eigendecomposition of the matrix $\mathbf{D}$ is given explicitly by the two sparse eigenvectors and their respective eigenvalues,

$$\mathbf{v}_{1,2} = \Big(\pm \frac{\sqrt{|\mathbf{d}_{kl}|^2 + |\mathbf{d}_{km}|^2}}{\bar{\mathbf{d}}_{km}},\ \frac{\bar{\mathbf{d}}_{kl}}{\bar{\mathbf{d}}_{km}},\ 1\Big), \quad \lambda_{1,2} = \pm\sqrt{|\mathbf{d}_{kl}|^2 + |\mathbf{d}_{km}|^2}.$$

with the vector entries being at positions k, l, and m. All remaining eigenvectors have vanishing eigenvalues, and thus do not contribute to the eigendecomposition. Setting $\mathbf{w}_i = \sqrt{|\lambda_i|}\mathbf{v}_i/\|\mathbf{v}_i\|$ and $\sigma_i = \operatorname{sign}(\lambda_i)$ for $i = 1, 2$, gives us the explicit vectors for the Cholesky updates.

If several matchings change at once, for instance, to move a branch point along a path of edges, then the Cholesky updates have to be performed one edge at a time.

Comparison

The timings in Table 6.1 show the speedup of the Cholesky updates in comparison to re-factorization of the matrix. To compute and update the Cholesky decompositions we are using the CXSparseJ library [Lin], a Java port of CSparse package by Tim Davis [Dav06], with support for complex matrices. The times were measured on an Intel Core i5-3570K processor.

Average times to re-compute E_D from scratch

model	f	min. $E(\alpha)$	compute curl$^*X(e)$ from α	decompose $\bar{\mathfrak{L}}^* = \mathbf{L}^{\mathsf{H}}\mathbf{L}$	solve $\mathbf{Lx} = \mathbf{b}$	other	total
bunny	3k	0.1 ms	0.2 ms	**15.5 ms**	0.1 ms	0.3 ms	**16.3 ms**
bunny	20k	1.0 ms	3.6 ms	**208.6 ms**	1.4 ms	0.7 ms	**215.5 ms**
armadillo	80k	4.1 ms	17.4 ms	**930.7 ms**	8.8 ms	3.7 ms	**965.0 ms**

Average times to re-compute E_D via Cholesky updates

model	f	min. $E(\alpha)$	compute curl$^*X(e)$ from α	update $\bar{\mathfrak{L}}^* = \mathbf{L}^{\mathsf{H}}\mathbf{L}$	solve $\mathbf{Lx} = \mathbf{b}$	other	total
bunny	3k	0.1 ms	0.2 ms	**0.1 ms**	0.1 ms	0.2 ms	**1.0 ms**
bunny	20k	0.9 ms	3.7 ms	**0.7 ms**	1.4 ms	1.7 ms	**8.7 ms**
armadillo	80k	4.1 ms	17.4 ms	**2.1 ms**	8.8 ms	8.0 ms	**40.5 ms**
armadillo	346k	26.8 ms	87.9 ms	17.4 ms	33.4 ms	62.5 ms	228.2 ms

Table 6.1: Times to compute the size of the curl part of $\tilde{X}$, with and without Cholesky updates.

The total computation time is sped up by a factor in the range around 20. To put it differently: We can update E_D for the 346kf armadillo model via Cholesky updates as fast as computing E_D for the 20kf bunny without the updates. When using Cholesky updates, the majority of time not even spent with solving and updating the linear systems, but with relatively trivial tasks like rotating the field by α and computing the corresponding curl at the edges.

6.2.2 A Curl Minimization Heuristic

Now that we have a fast way to compute the E_D, we can improve the initial branch point layout by changing the matching at an edge, and re-evaluate E_D in hope to find an improvement. We start off with the smoothed frame field that we received from the extension of directions in the trusted regions and its resulting matching, which is computed as in Section 5.1.

We have experimented with many approaches to improve the branch point layout, and we have found that the following ingredients are essential for successful optimization.

Cancellation. As noise might cause a lot of branch points, the first quick step is to eliminate misplaced pairs of branch points, whenever this causes E_D to decrease. The pairs are found by breadth first search from each branch point to some fixed number k of nearby branch points with opposite sign of index. If the movement of p to one of the k neighbors decreases E_D, the pair is eliminated.

Branch point movement and creation. As a main loop, we change one matching at a time, to check if E_D is decreased. This action might eliminate or create a pair of branch points, or, more often, move a branch point to an adjacent vertex (of course, there are more operations involving branch point indices different from ± 1, but these branch points appear rather rarely). It is very important to allow also the changes of the matching, which lead to the *creation* of new branch points, as E_D can be reduced *significantly further*, if new branch points are allowed. While the movement of existing branch points has been discussed in the literature [Nie12, BZK09, MZ13], the importance of *new* branch points has not gotten much attention. However, if new branch points are unwanted, we simply skip those changes which would increase the branch point count.

Sorting. The number of edges on which we could change the matching is very high (around $\mathsf{v} + \mathsf{f}$), so it is crucial to prioritize promising candidates for changes of the matching. We found that the most promising candidates are those edges, who contribute most to the total error E_D. As $E_D = \frac{1}{2}\mathbf{v}^{\mathsf{H}}\,\overline{\mathfrak{L}}^*\,\mathbf{v} = -\frac{1}{2}\sum_e \mathbf{v}_e \cdot \mathrm{curl}^* X(e)$, we can associate the share of the total distortion to each edge. Sorting the edges is crucial. We give a visual comparison how fast E_D is reduced with and without sorting in Figure 6.1. Note, that we actually have to compute $\mathbf{v}$ via forward and back substitution in $\mathbf{L}^{\mathsf{H}}\mathbf{L}\mathbf{v} = \mathbf{b}$, whereas the computation of $E_D = \frac{1}{2}\mathbf{x}^{\mathsf{H}}\mathbf{x}$ requires only the forward substitution $\mathbf{L}\mathbf{x} = \mathbf{b}$.

Balancing. While there is a lack of attention on the creation of branch points, there is also a lack of focus on finding the right number of branch points. Section 6.4 is dedicated to the right balance of the number of branch points, and we will propose a method to eliminate excessive branch points there. We summarize our ingredients in the following algorithm.

Algorithm 6.1 (Curl reduction)

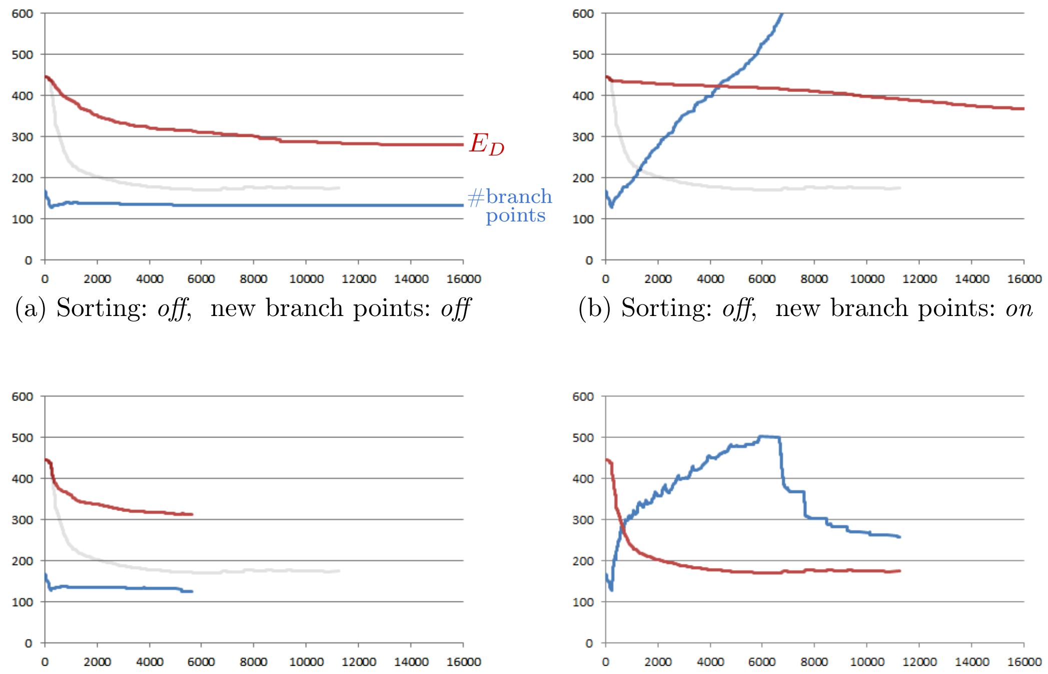
```
1. (Cancellation) For each branch point $p$:
   Move $p$ to one of the $k$ nearest branch points to $p$, if this decreases $E_D$.
2. (Sorting) Sort the edges by their share of the distortion, $|\mathbf{v}_e \cdot \mathrm{curl}^* X(e)|$
3. (Curl reduction) For each edge $e$ (in decreasing order of the edge's distortion),
   (a) Set $m_e \leftarrow m_e - \mathrm{sign}(\theta_e)$ if this decreases $E_D$.
   (b) If more than 1‰ of matchings changed in this reduction loop, go to Step 2.
4. (Balancing) Cancel insignificant branch points:
   (a) For each branch point, find the $k$ nearest candidates for canceling.
   (b) Measure the increase $\Delta_{E_D}$ of $E_D$, that each cancelation in 4(a) would cause.
   (c) Cancel the branch point candidates in increasing order of $\Delta_{E_D}$ if the impact
       on $E_D$ is below expected continuity error reduction $\Delta_{E_H}$ (see Section 6.4).
   (d) If branch points were canceled in Step 4, go to Step 2, but disallow any new
       branch points from now on.
```

Discussion

We visualize the distortion reduction over time in Figure 6.1, using the casting model with 10224 faces. Two results are shown there: First, the creation of branch points, helps to push E_D *significantly* further down, and second, *if* new branch points are allowed, then edge sorting is vital.

The effects of the branch point relocation is shown in Figures 6.7 and 6.8 in the results section. We observed that changing the matching by $-\mathrm{sign}(\theta_e)$ in Step 3(a) works slightly better than trying both possible signs, even though the sign of θ_e is only a weak hint of the direction in which a change of the matching decreases E_D. The constant k in steps 1 and 4 does not have to be very large. We usually use $k = 3$, as higher values significantly slow down the process without contributing much to better results.

The (local) minimizers of E_D involve a high number of branch points, much higher than the numbers of branch points which are found by our frame field extension method or by state-the-art algorithms for frame field generation. Especially those algorithm



(a) Sorting: *off*, new branch points: *off* (b) Sorting: *off*, new branch points: *on*

(c) Sorting: *on*, new branch points: *off* (d) Sorting: *on*, new branch points: *on*

Figure 6.1: The graphs show the development of the distortion E_D (red curves) and the number of branch points (blue curves, y-axis) over the number of iterations, using our heuristic with different settings: Sorting is turned off in the top row, and new branch points are prohibited in the left column. The light gray curve in (a)-(c) is the distortion curve of (d), for comparison. The first 230 iterations in all cases were spent for *cancellation*. In (d), iterations 6000 to 11000 were spent for *balancing*.

which focus on minimizing the field's smoothness (such as holonomy angles) create rather few branch points, leading to sub-optimal parameterizations.

If we do allow new branch points, and test edges in arbitrary order, then the convergence is unfeasibly slow and creates an excessive amount of branch points (the branch point curve in Figure 6.1(b) continues to rise steeply!). The excess is much lower if we sort the edges: The distortion tends to be concentrated around branch points, and so it is very likely that a change of matching moves a branch point instead of creating a branch point pair if we sort the edges by distortion. Besides the excessive creation of branch points, also the error reduction speed is unfeasibly slow when edges are not sorted by significance. Without sorting, the energy is reduced by 9.1% within the first 10 000 iterations. If the edges are sorted, the same reduction is achieved within only 59 iterations!

The branch point curve Figure 6.1(d) shows that we let the number of branch points rise significantly, and then we cancel out the insignificant ones. While this might seem awkward (because we could immediately skip the creation of branch points that barely reduce E_D), it is necessary in our heuristic: While branch point pairs might reduce E_D only slightly when one edge apart, the two branch points might move further apart during the optimization and gain significance. Another common scenario is that rows of branch points of alternating index are created, and intermediate branch points are later cancelled which were not created together as a pair.

It would of course be very helpful to be able to compute a gradient of E_D in terms of the matching. Surprisingly, we observed rare cases in which the distortion E_D was reduced by changing the matching at some edge e, *no matter if m_e was increased or decreased by 1*, in contradiction with the idea of a gradient, which should deliver a direction of the energy's descent. The cases in which this effect appeared always involved extremely high distortion at e that caused the local parameterization function to flip its orientation in one of the adjacent elements. Moreover, I believe that the non-monotonicity effect at e is caused by the periodicity of the edge curl with respect to the curvature θ_e of the frame field at e.

Comparison

Just recently, Myles and Zorin [MZ13] showed that the as-rigid-as-possible (ARAP) metric distortion (which is implicitly included in E_D) can be pushed arbitrary low, if only the surface can be triangulated fine enough. They observed the appearance of cone chains, similar to our results in Figures 6.7(d) and 6.8(d). However, Myles and Zorin do not cover the creation of quadrilateral meshes under the existence of cone chains, because they did not ensure the parameter line continuity condition.

In our examples, we show that these cone chains can be applied successfully to generate low distortion meshes. The cone chains are not placed explicitly, but they appear automatically during the minimization of curl in Algorithm 6.1. The balancing phase in Algorithm 6.1 makes sure that the number of branch points in the cone chains stays under control. With these measures, the total metric distortion in our parameterization can be kept significantly below that of Myles and Zorin, compare Figure 6.2.

6.3 The Minimization of Rounding Errors

Section 2.4.2 and Section 4.2.1 describe the rounding process to get a visually continuous parameterization: From the computed periods $\mathbf{b}$ of the frame field X, we compute the closest integer vector $\mathbf{c} = [\mathbf{b}]$ by rounding the entries of $\mathbf{b}$.

Then we add the harmonic field $H_\mathbf{d}$ with periods $\mathbf{d}_i = \mathbf{c}_i - \mathbf{b}_i$ to X. Because this procedure alters our input field by $H_\mathbf{d}$, the numbers $\mathbf{d}_i$ should be chosen so that $\|H_\mathbf{d}\|_2$

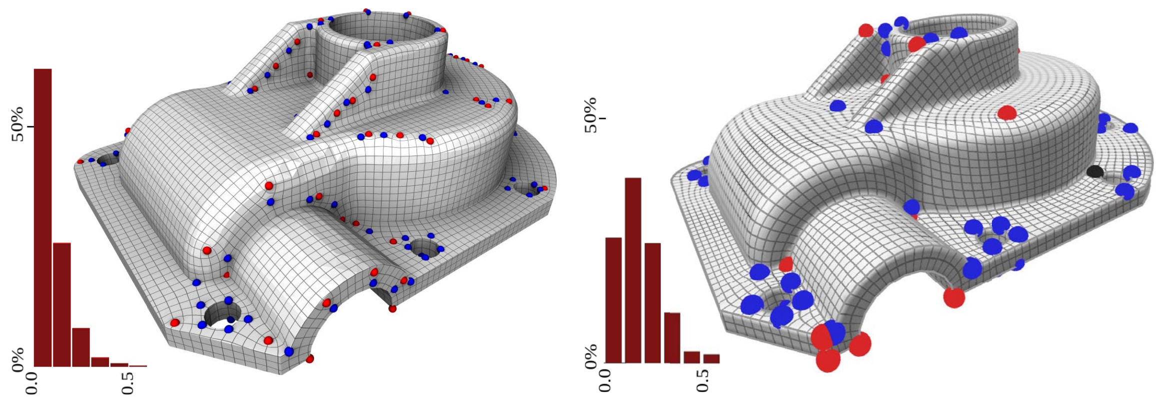


Figure 6.2: Our result (left) compared to the very recent work of Myles and Zorin [MZ13] (right, image source: Myles and Zorin). The histograms show the distribution of the as-rigid-as-possible distortion over the individual triangles, as used by Myles and Zorin. The average ARAP distortion per triangle of our (their) parameterization is 0.110 (0.211), and 95% (86%) of the triangles have a distortion below 0.3. Note that is is not quite an apples to apples comparison: While we do not respect hard constraints to keep parameter lines exactly at sharp edges, we respect more soft constraints than their parameterization (*i. e.*, parameter also follow rounded edges). Since our parameter lines are always very close to the sharp edges, I conjecture that the impact of hard constrains would not be substantial.

is small, which turns out to be a fairly hard optimization problem. In this section, we forego the complex notation for the period vectors, and use real-valued vectors, $\mathbf{b}_0 = \bar{\mathbf{b}}_0$, $\mathbf{b}_1 = \boldsymbol{i}\bar{\mathbf{b}}_0 \in \mathbb{R}$, *etc.*, instead.

Since $H_\mathbf{d}$ is a linear combination $\sum_i \mathbf{d}_i H_i$ of harmonic basis fields H_i, we have

$$E_H := \tfrac{1}{2}\|H_\mathbf{d}\|_2^2 = \tfrac{1}{2}\mathbf{d}^\intercal \mathbf{P}\mathbf{d} = \tfrac{1}{2}(\mathbf{c} \quad \mathbf{b})^\intercal \mathbf{P}(\mathbf{c} \quad \mathbf{b}),$$

with a real matrix $\mathbf{P}$ that contains the scalar products of the harmonic fields,

$$\mathbf{P}_{ij} := \int_M \langle H_i, H_j\rangle \mathrm{d}A.$$

The dimension of the square matrix $\mathbf{P}$ equals the number of symmetric vector fields on the covering surface, which is $\tilde{\mathsf{h}}_{\mathrm{sym}} = 4\mathsf{g} + 2(\mathsf{b}_{\mathrm{even}} + \mathsf{b}_{\mathrm{odd}}) \quad 4$, according to (4.1).

Finding a vector $\mathbf{d}$ which minimizes $\mathbf{d}^\intercal\mathbf{P}\mathbf{d}$ is equivalent to the closest vector problem (CVP), which asks, given a rational lattice matrix $\mathbf{B}$ and a rational target vector $\mathbf{t}$, for an integer vector $\mathbf{x}$ so that $\|\mathbf{B}\mathbf{x} \quad \mathbf{t}\|$ is minimized, see [MG02]. By setting $\mathbf{B}$ to the Cholesky factor of $\mathbf{P}$ we get $(\mathbf{c} \quad \mathbf{b})^\intercal\mathbf{P}(\mathbf{c} \quad \mathbf{b}) = \|\mathbf{B}\mathbf{c} \quad \mathbf{B}\mathbf{b}\|^2$, so that the equivalence of the two problem formulations becomes obvious. Even though CVP is NP-hard, it does not necessarily mean that our optimization problem is NP-hard as well, because our matrix **B** has not arbitrary form (in particular, **B** is a triangular matrix, and it is not clear if the fact that **P** is a period matrix adds any restrictions to **P** besides making it positive semidefinite). Nevertheless, we could not find any hint that the restrictions simplify the problem.

6.3.1 Minimizing $(\mathbf{c}-\mathbf{b})^\intercal\mathbf{P}(\mathbf{c}-\mathbf{b})$

The fact that $(\mathbf{c}-\mathbf{b})^\intercal\mathbf{P}(\mathbf{c}-\mathbf{b})$ is hard to minimize does not reduce our need for a good solution, but it suggests that brute-force search is a reasonable strategy. If we modify the i-th entry of $\mathbf{c}$ by $\sigma_i = \pm 1$, the value of $\|H_\mathbf{d}\|_2^2$ changes by

$$\delta = \mathbf{P}_{ii} + 2\sigma_i \mathbf{P}_i^\intercal(\mathbf{c}-\mathbf{b}),$$

where $\mathbf{P}_i$ is the i-th column of $\mathbf{P}$. If δ is negative, we have improved our solution and the modification of $\mathbf{c}$ should be maintained. Thus, we iteratively check all indices of $\mathbf{c}$ for improvement and stop if δ is non-negative for all i.

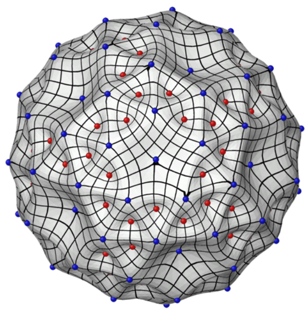
It is unlikely that the optimal vector $\mathbf{c}$ is found by changing one entry at a time. Carrying the idea further, one can change r entries at a time and compute δ via

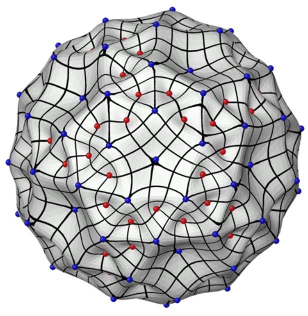
$$\delta = \sum_{\sigma_i \neq 0} \Big(p_{ii} + 2\sigma_i\, \mathbf{P}_i^\intercal(\mathbf{c}-\mathbf{b}) \Big) \; + \sum_{\sigma_i\sigma_j \neq 0} \sigma_i\sigma_j p_{ij},$$

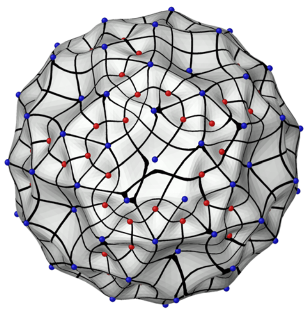
where $\delta \in \{-1,0,1\}^{\tilde{\mathsf{h}}_{\text{sym}}}$ has at most r non-zero entries. Of course, testing $O(\tilde{\mathsf{h}}_{\text{sym}}^r)$ index combinations drives the computational complexity quickly up. However, if the dot products $\mathbf{P}_i^\intercal \cdot (\mathbf{c}-\mathbf{b})$ are pre-computed in the beginning, each computation of δ can be performed with very few operations, independent of the size of $\tilde{\mathsf{h}}_{\text{sym}}$. For example, if $r = 3$, then 20 basic operations are sufficient to compute δ. Only if the current solution is improved, the pre-computed dot products must be updated.

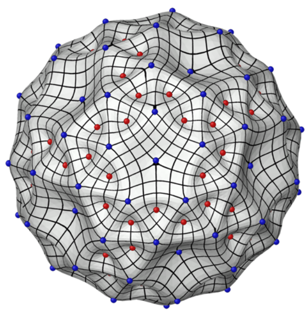
Testing a single configuration of $\mathbf{c}$ is so fast, that even billions of combinations can be checked in short time. We found that using $r = 3$ is practical for up to several hundreds of branch points.

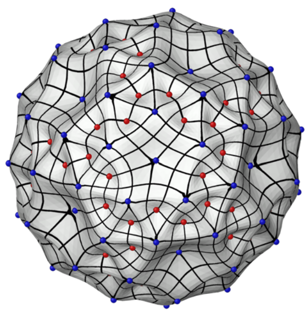
Classical approximation algorithms of the CVP problem are usually based on Lenstra-Lenstra-Lovász lattice basis reduction. The LLL-algorithm [LLL82] finds a nearly orthogonal lattice basis **B**, which increases the probability of finding good vectors **c**. Even though the LLL-algorithm can be performed relatively quick on triangular basis matrices **B**, we found that the basis reduction is not worth the effort. The matrix **B** is already relatively well conditioned (which in parts due to the use of a *shortest* cut graph through the branch points), and we could not observe that better solutions are found with a reduced basis.


(a) $E_H = 2.86\%$


(b) $E_H = 5.95\%$


(c) $E_H = 8.64\%$


(d) $E_H = 1.17\%$


(e) $E_H = 3.50\%$

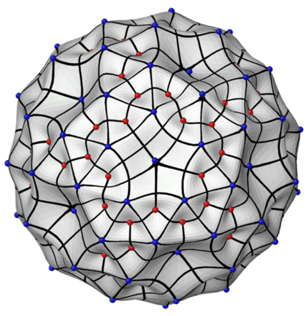

(f) $E_H = 3.35\%$

FIGURE 6.3: The importance of good rounding increases with the number of branch points and the coarseness of parameter lines. *Top row*: straightforward rounding. *Bottom row*: optimized rounding according to Section 6.3.1. The amount of distortion, $E_H = \frac{1}{2}\|H_{\mathbf{d}}\|^2$, is given in relation to the size of the input field, $\frac{1}{2}\|X\|^2$. The error reduction using $r = 3$ is 41 to 61% on these examples, and averages around 40%–45% on larger data sets. It took less than 4 seconds to compute the period matrix and check more than one hundred million rounding possibilities.

6.3.2 COMPUTATION OF THE PERIOD MATRIX

Formally, the matrix $\mathbf{P} = \left(\int_M \langle H_i, H_j \rangle \mathrm{d}A\right)_{i,j}$ is easy to describe, but its naive computation fails even for medium-sized models: Computing a harmonic field H_j on the fly for each of the $\tilde{\mathsf{h}}^2_{\text{sym}}$ entries of $\mathbf{P}$ is just as prohibitively expensive as storing all $\tilde{\mathsf{h}}_{\text{sym}}$ harmonic fields in memory to save the recomputation. These two options either require too much computation time or too much space in memory, even if all fields H_i can be found with the same Cholesky factorization.

Recall the vector field Σ_i from (1.13) which has local support to the left of the i-th homotopy basis path γ_i, and whose periods match those of H_i. Because $\Sigma_i - H_i \in \mathcal{P}$ and $\mathcal{P}$ and $\mathcal{H}$ are mutually orthogonal, the scalar products $\langle H_i, H_j \rangle$ and $\langle H_i, \Sigma_j \rangle$ coincide. (The value of $\langle H_i, \Sigma_j \rangle$ corresponds to the through flow of H_i through γ_j).

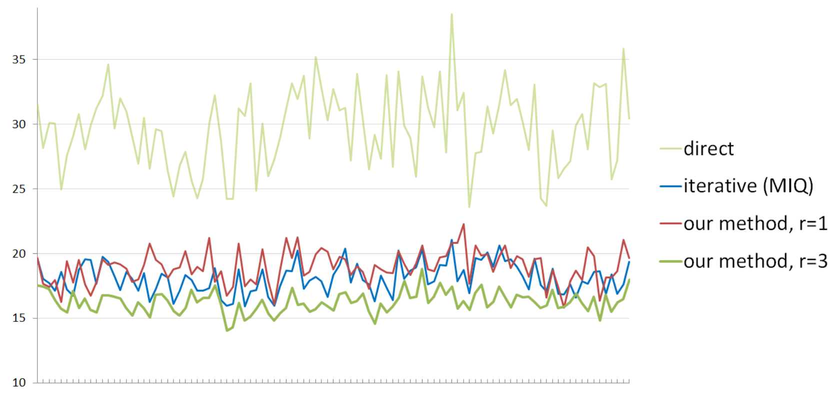


Figure 6.4: The performance of rounding strategies. The curves show the error E_H over 100 instances of the rounding problem. *Light green:* direct rounding, *Blue:* iterative rounding as used in MIQ, *Red:* our method using $r = 1$, *Green:* our method using $r = 3$. Note that the y-axis is not zero-based.

Thanks to the local support of Σ_j, it is no problem to store all Σ_j in a sparse data structure. Therefore, if the locally supported fields $\Sigma_1, \ldots, \Sigma_{\tilde{\mathsf{h}}_{\mathrm{sym}}}$ are pre-calculated, only one harmonic field H_i must be computed to fill the i-th column of $\mathbf{P}$. All entries in this column can be computed via $\langle H_i, \Sigma_j \rangle$. Still, solving for H_i is then the most time consuming part in the computation of $\mathbf{P}$.

6.3.3 Comparison

Before we implemented the computation of the period matrix, we used to change one entry of $\mathbf{d}$ at a time, and recompute and measure the harmonic field $H_{\mathbf{d}}$. The rounding performance is then equivalent using a recursion depth of $r = 1$ in the above method, however, our new method is faster, because every harmonic field is computed only once per entry of $\mathbf{d}$. Similarly, Springborn *et al.* also modify one entry at a time in their Conformal Equivalence paper [SSP08], Section 6.1. However, since they did not specify any geometric norm (*i. e.*, $\mathbf{P}$) to use, I suppose that they used a purely combinatorial one (that depends only on the connectivity of the cut graph or *meta polygon*).

The Mixed Integer Quadrangulation (MIQ [BZK09]) method starts without any integer constraints, and iteratively fixes one variable to to an integer value. In each iteration, the variable is fixed which causes the smallest increase of their energy. Although used in a slightly different context, the same concept can be also used for our rounding problem.

To compare the methods with each other, we generated one hundred test instances on the bunny model (as we will explain in Section 6.4, the characteristic of the period matrix $\mathbf{P}$ does not change with the model, so we confine ourselves to one test model). We compared the iterative rounding as used in MIQ to our method using $r = 1$ and

$r = 3$, and got the following average results, normalized to the non-optimized case.

● direct rounding	● iterative greedy rounding (MIQ)	● our method $r = 1$	● our method $r = 3$
100%	61.0%	63.8%	54.8%

The results of the individual test cases are shown graphically in Figure 6.4. The comparison shows that it is crucial to have some rounding strategy, but also that our proposed method, using $r = 3$, can push the error much further than the other methods. While the difference of 10% and 14% smaller rounding errors versus MIQ and $r = 1$ are not ground shaking, they are noticeable and the benefits come essentially for free: Just like for the MIQ and $r = 1$ case, one linear system has to be solved per variable, which is the slow part (where MIQ fixes variables, and we solve pre-factorized systems). Once that is accomplished, the evaluation E_H is so fast, that it would be wasteful not to compute more rounding possibilities.

6.4 Balancing Curl and Rounding Errors

We have seen in Section 6.2 that the curl distortion E_D can be drastically reduced if new branch points are introduced. In contrast, the global rounding problem has to consider more constraints when the number of branch point rises, and so, the distortion E_H grows with the number of branch points. In Figure 6.5 we plotted the two distortion measures E_D and E_H over the number of branch points for two models whose line density is as shown on the models. For every surface, the number of branch points has a sweet spot which minimizes the total distortion $E_H + E_D$, in dependence of the parameter line density.

To find this sweet spot, we estimate E_H in dependence of the number of branch points as explained below. With an estimate of E_H we are then able to predict if the cancellation of two branch points—causing a certain increase of E_D—will lower the total distortion $E_D + E_H$. The balance between the two errors E_D and E_H via the estimation of E_H is the key to Step 4 of Algorithm 6.1, and that is why we called this step *balancing*.

Note that if we scale the input field X by a factor a, then E_D will grow with the square of a, whereas the distortion E_H is expected to keep the same magnitude, as the integer remainders which determine E_H always stay in the range $[-0.5, 0.5]$, independently of the scale. Therefore, if the line density is high, *i. e.*, the factor a is large, the sweet spot moves towards a high number of branch points. In turn, if the line density is low, $E_D + E_H$ will favor less branch points.

The Average Cost of a Branch Point: An Estimation of E_H

For Step 4, we found that it is sufficiently accurate to assume that E_H rises linearly with the number of rounding conditions, which is roughly twice the number of branch points. The green curves in Figure 6.5 suggest that once we measure E_H at one point of the curve, we can pretty well estimate the E_H for some other number of branch points by linear extrapolation.

When having a closer look at the continuity distortion E_H, it is not surprising that it rises linearly with the number of branch points. Recall that $E_H = \frac{1}{2}\mathbf{d}^\mathsf{T}\mathbf{P}\mathbf{d}$. Without any further knowledge about the surface geometry, the initial, unoptimized entries of d are expected to be distributed independently, symmetrically and (almost) uniformly in $[-0.5, 0.5]$. The expected value, $\mathrm{E}[E_H]$, of E_H, is thus

$$\mathrm{E}[E_H] = \frac{1}{2}\sum_{i,j}\mathrm{E}[\mathbf{d}_i\mathbf{d}_j]\mathbf{P}_{ij} = \frac{1}{2}\sum_{i}\mathrm{E}[\mathbf{d}_i^2]\mathbf{P}_{ii} = \frac{1}{24}\mathrm{tr}(\mathbf{P}),$$

since $\mathrm{E}[\mathbf{d}_i\mathbf{d}_j] = \frac{1}{12}$ for $i = j$, and 0 otherwise, if $\mathbf{d}_i$ is uniformly distributed. Only if the input field is scaled extremely small, the entries of the period vector, $\mathbf{b}_i$, would cluster around zero and so would $\mathbf{d}_i = \mathbf{b}_i - [\mathbf{b}_i]$. The only thing that changes in this case is that the factor of $\frac{1}{12}$ would have to be replaced by a smaller value of $\mathrm{E}[\mathbf{d}_i^2]$.

If we assume that $\mathrm{tr}(\mathbf{P})$ grows linearly with the size of $\mathbf{P}$, it means that all diagonal entries $\mathbf{P}_{ii} = \|H_i\|^2$ must be of similar magnitude. This is indeed to be expected: The harmonic fields H_i are mostly concentrated around the branch points which are connected by their respective homology generator γ_i which goes from one branch point to another. The area in which H_i has meaningful size grows with the length of γ_i, but at the same time, the magnitude of H_i shrinks with the length of γ_i. Because the two effects cancel each other out, $\|H_i\|^2$ does not depend directly on the length of γ_i. Instead, the norm of H_i is mainly influenced by the shape and discretization of M around the branch points, and has mostly values around 2 to 3.

We measure the distortion error E_H after the curl reduction step, because at this stage we have many branch points to level out the influence of random values. Once we measured E_H, we use

$$\Delta_{E_H} := \frac{E_H}{2\tilde{\mathsf{h}}_{\mathrm{sym}}}$$

to compute the expected continuity distortion per pair of (complex) rounding constraints and use it as a fixed value.

Balancing

With an estimate of the continuity distortion, we can be more detailed on the *balancing* step of Algorithm 6.1. The *curl reduction* step usually creates a lot of new branch

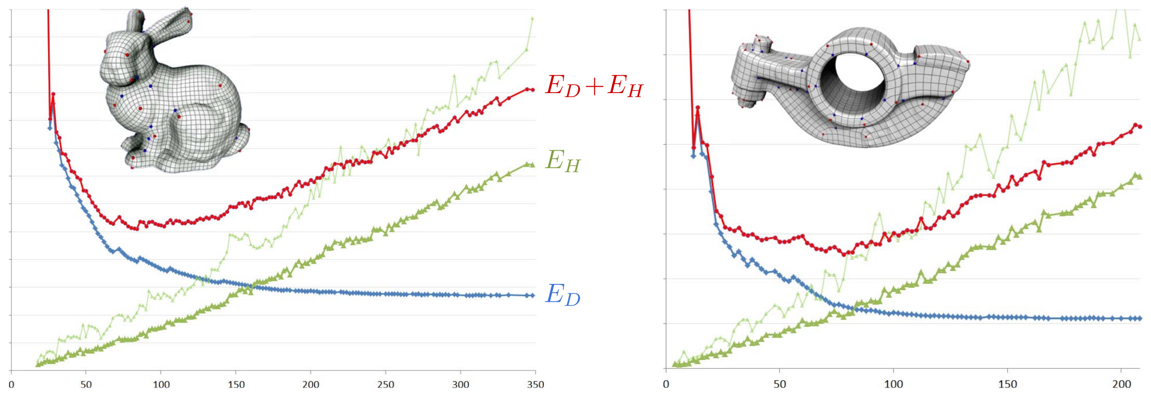


Figure 6.5: Each model has an optimal number of branch points with respect to a given line density. While the distortion of the curl removal, E_D, can be lowered with more branch points, additional branch points will increase the distortion enforced by the global continuity conditions, E_H. The graphs show the distortion E_D in blue, E_H in green as well as their sum in red over the number of branch points. The light green curve represents E_H before optimization.

points and at the end of the curl reduction step, the expected continuity distortion might dominate over the curl distortion, as seen on the right end of the graphs in Figure 6.5.

If we were to cancel a pair p, q of branch points of opposite index, we can quickly compute the increase $\Delta_{E_D} = \Delta_{E_D}(p, q)$ of E_D via the updates of the Cholesky factorization, and decide if the cancelation of p and q pays off by checking if Δ_{E_D} is smaller than the estimated decrease Δ_{E_H}.

Ideally, we would compute the impact of E_D for all pairs p and q worth considering, and cancel the branch point pair which has the lowest value of Δ_{E_D}. Because this would require the re-computation of Δ_{E_D} for the cancelation of each single pair of branch points, this is definitively too slow. Since most of the computed values of Δ_{E_D} change only marginally upon the cancelation of one branch point pair, we compute all the cancelation costs only once in step $4(b)$ for sorting, and cancel all pair branch point pairs for which $\Delta_{E_D} < \Delta_{E_H}$ in this ordering. Before performing the cancelation, however, we must check if the increase of E_D is still below Δ_{E_H}, because the matching—and thus E_D—might have changed in the meantime, or one of the branch points might not exist anymore, if it was canceled otherwise.

Once we looped through all cancelation candidates with $\Delta_{E_D} < \Delta_{E_H}$, we return to Step 2 of the algorithm to re-position the branch points which were influenced by the branch point canceling. From now on, we disallow any new branch points, to avoid repeated creation and canceling of the same branch points. As a side effect, the curl reduction loop will be much faster now, because only edges at branch points have to be checked for improvements.

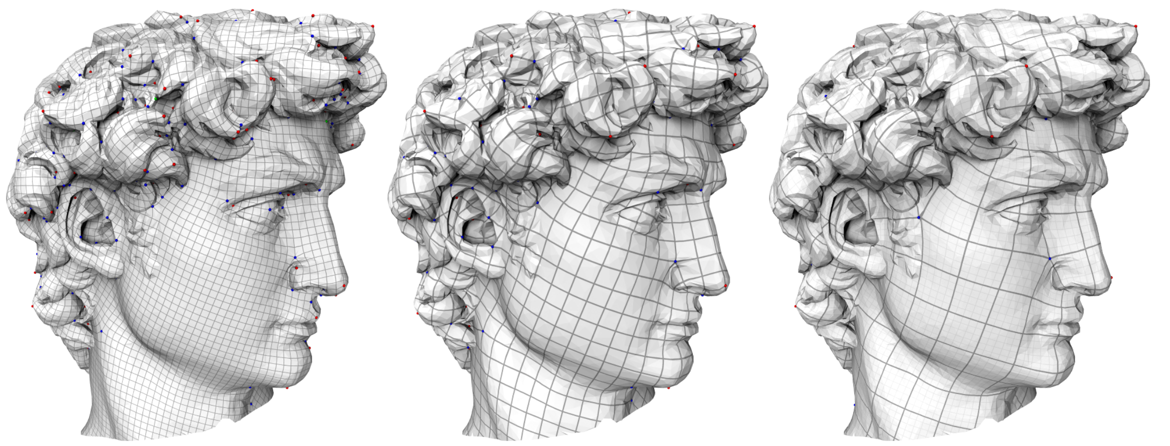
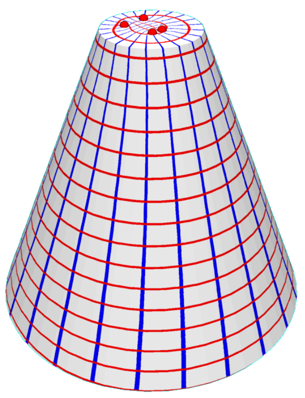
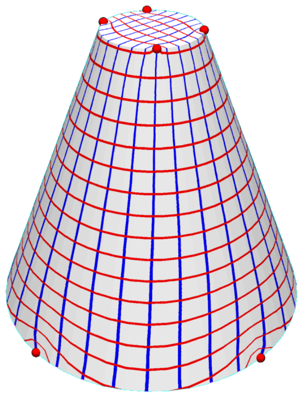
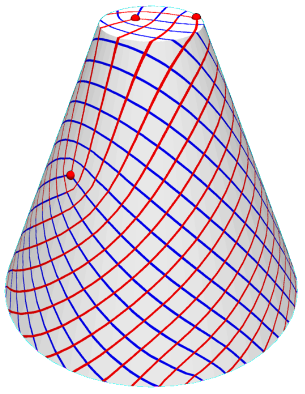

467 branch points
$E_D = 5.74$, $E_H = 5.11$
153 branch points
$E_D = 7.08$, $E_H = 6.53$
28 branch points
$E_D = 9.30$, $E_H = 5.22$

Figure 6.6: The head of Michelangelo's David statue is parameterized with several parameter line densities. To counter increasing continuity distortion with higher coarseness of the grid, the algorithm cancelled significantly more branch points in the balancing phase for the coarse parameterizations.

When returning to the balancing phase again, we update the list of canceling candidates and re-measure Δ_{E_D}, so that new candidates will again satisfy $\Delta_{E_D} < \Delta_{E_H}$, and we iterate this process of branch point merging and relocation until no cancelation with $\Delta_{E_D} < \Delta_{E_H}$ is found. In the example of Figure 6.1, we can see six drops in the branch point curve caused by the balancing step. Each drop in branch points is accompanied by a small peak of E_D, which is then partly compensated by branch point repositioning in the curl reduction step.


(a) Basic QuadCover, using principal curvature directions and smoothing via Equation (5.3). Parameter lines on the top are very dense and curved. Although its symmetry looks very aesthetic, the distortion (*i. e.*, deviation from isometry) is rather large.


(b) Branch points were relocated to reduce the smoothness energy $E(\alpha)$. Parameter lines on top of the cone are much straighter, but horizontal parameter lines are highly distorted by trying to stay equidistant throughout the top and bottom. Length distortion is still large.


(c) Algorithm 6.1 is applied to (b) to reduce curl part of the frame field, but without allowing it to create new branch point pairs. One branch point moved to the lateral surface. Alignment to curvature directions is turned off.

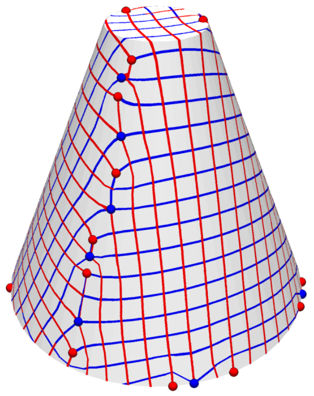
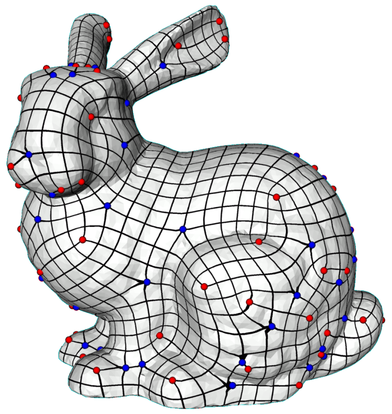
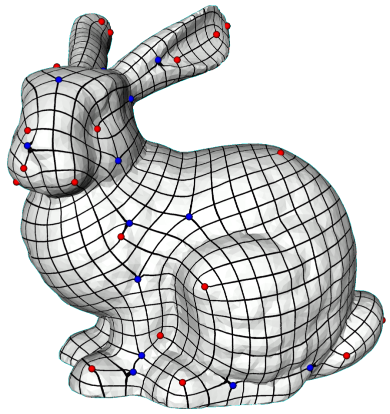
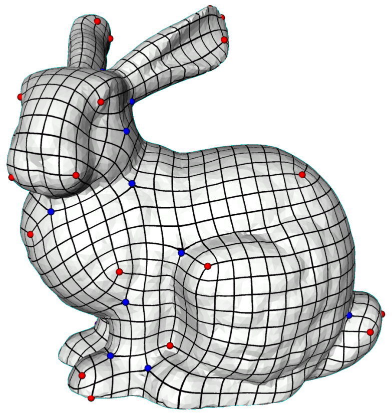

(d) Algorithm 6.1 is allowed to create new branch point pairs. This results in a large patch of extremely low distortion, "sewn together" at a lines of branch points. This effect is also visible in Figure 6.8(d).

Figure 6.7: What is a regular parameterization of a truncated cone? Its shape requires that alignment to curvature necessarily comes at the cost of high length distortion (top row). The generation of unit size quads requires branch points on the lateral surface (bottom row), or parameter lines would otherwise be either condensed at the top or stretched at the bottom. The cone example shows that a *best* parameterization can only be determined with respect to a certain application.


(a) Basic QuadCover, using principal curvature directions and smoothing via Equation (5.3).


(b) Propagating curvature directions from regions with high curvature stability leads to fewer branch points and more regular parameterizations (Section 5.1.2).


(c) Branch points of (b) are moved to reduce curl distortion E_D (Section 6.2), but no new branch points were permitted. The reduced curl leads to very high angle and length regularity, while lines follow the principal curvature directions at the same time.

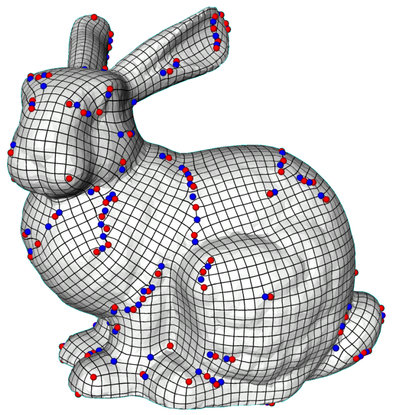

(d) Algorithm 6.1 is allowed to create new branch point pairs, compare Figure 6.7(d), To handle the high branch point count, the line density is increased. Distortion is extremely low while curvature directions are preserved.

Figure 6.8: The figure shows the effects of various frame field optimizations applied to the Stanford bunny model.

Chapter 7

Results and Conclusions

QuadCover

With QuadCover, we have developed a well-received algorithm for surface parameterization, and we could improve direction-guided surface parameterization and quadrilateral remeshing over what was available at that time.

Even though I consider QuadCover as one of the main contributions of this thesis, I confine myself to use only the comparison of results from the original paper in Figures 7.1 and 7.2. I tried to not repeat the QuadCover article here, but instead to emphasize its details and extensions that did *not* appear in the original publication. The parts of QuadCover that turned out to be very effective—such as the Hodge decomposition, and the use of covering spaces as a theoretical support—are described in much greater detail than in the original paper. Some QuadCover details, however, could be replaced by newer, more effective ideas. This includes, for example, the smoothing of frame fields in Section 5.2, or shorter cut paths that we optimized in [KNP10].

To show the performance of QuadCover and its improvements since its publication, we include Table 7.1 that appeared in the QuadCover article. It expresses the regularity of parameterizations by the relative standard deviation (RSD) of edge lengths and vertex angles. We extended the table by our current results, using only branch point relocation ('Relocation') as well as full pipeline of branch point creation, relocation, and balancing as in Algorithm 6.1 ('Balancing').

	[RLL$^+$06]	[TACSD06]	[DBG$^+$06]	QuadCover	Relocation	Balancing
vertices	6355	6576	7202	6535	6464	6506
branch points	314	34	26	37	59	93
RSD edge	25.0%	28.3%	30.8%	18.2%	9.0%	8.1%
RSD angle	10.7%	12.6%	7.8%	14.8%	8.0%	7.2%
ARAP energy	?	?	?	0.50	0.36	0.33

Table 7.1: Regularity metrics of the bunny models of Figure 7.1.

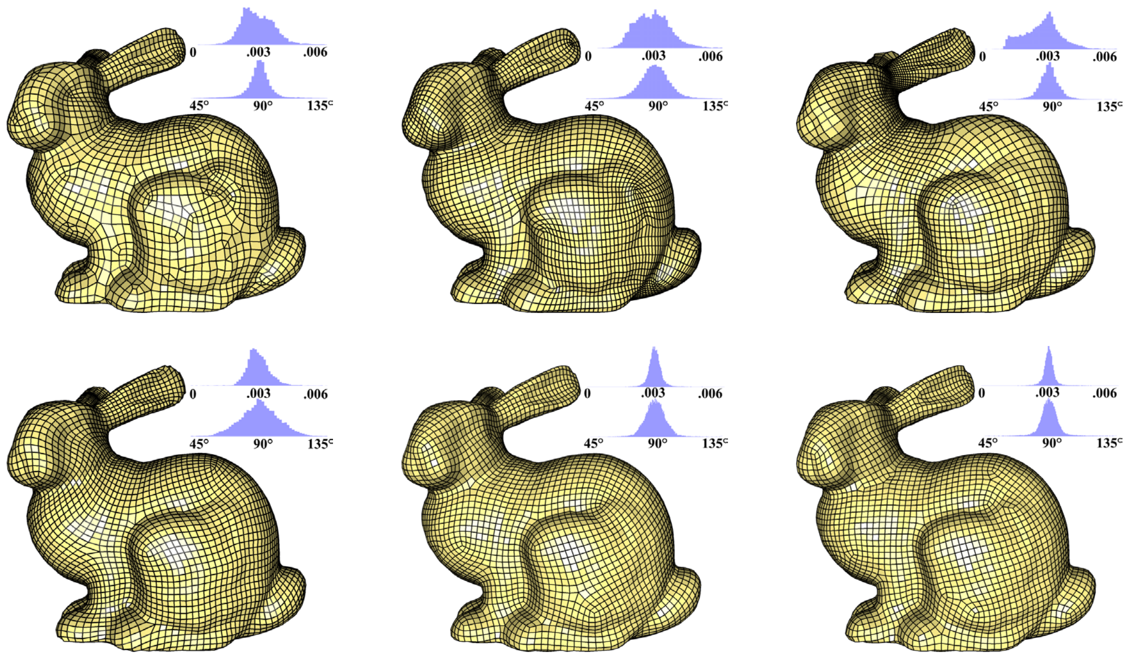


Figure 7.1: Comparison of remeshing results of the Stanford bunny. Models in the top row were produced by [RLL+06], [TACSD06], and [DBG+06]. In the bottom row we used QuadCover with different settings. Left: QuadCover as in [KNP07], middle: branch point relocation to minimize E_D, right: branch point creation, relocation, and balancing as in Algorithm 6.1. The histogram next to each model shows the distribution of edge lengths, the lower histogram represents angle distribution.

Apart from the numbers of table 7.1, we refer to more than a dozen examples, in this thesis which show that our method works well on various surfaces.

While some QuadCover improvements in this thesis, like the use of complex notation, are of rather cosmetic nature, the suggested optimizations of Chapter 6 increase the quality of QuadCover parameterizations significantly. The three main ingredients—curl reduction, rounding error reduction, and balancing the number of branch points—act independently and can be used individually in other parameterization systems.

Low Curl Branch Points

The relocation of branch points to improve the parameterization regularity has been used before [BZK09, KP10, Nie12], yet it has not been fast enough to make it practically useful. With Cholesky updates, the relocation can be finally performed in reasonable time. But even if one manages to circumvent the pitfalls of a speedup via sparse Cholesky updates, we showed that brute force optimization is not practical until a reasonable edge ordering is used.

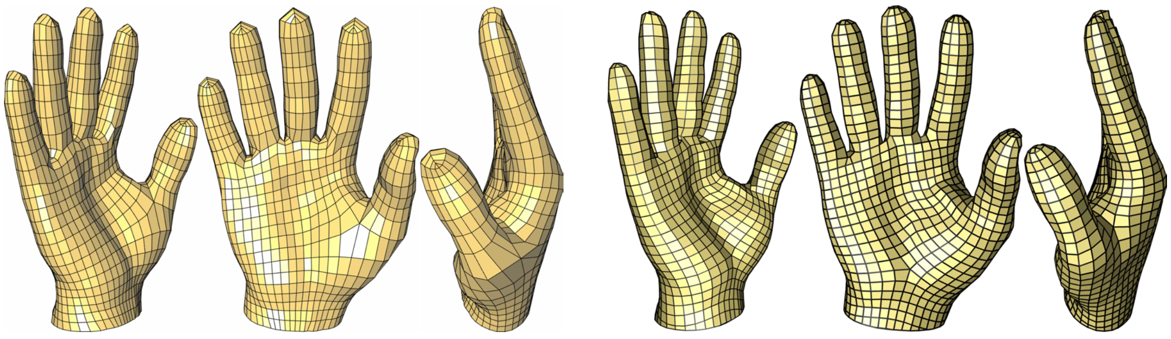


FIGURE 7.2: Comparison of our original QuadCover method (right) versus [TACSD06], which was state-of-the-art at the time of publication. In contrast to the latter, QuadCover needed no manual interaction besides setting setting preprocessing parameters.

We found that the branch point relocation works so well that many pre- and post-processing steps in our software are made redundant. For example, we have used two-step branch point relocation for a long time in our software: First relocate branch points to minimize the smoothing energy (as we did in Figure 6.8(b)), and then relocate branch points to minimize the curl (Figure 6.8(c)). This two-way minimization was necessary because fast relocation was not available.

An common post-processing problem is heavy local distortion and foldovers around branch points. By branch point relocation and branch point balancing, these artifacts can are drastically reduced, so that counteractive measures, such as the local stiffening of MIQ, are often unnecessary.

Better Rounding

We showed that by pre-computing the period matrix, the rounding error E_H can be drastically reduced. Our method performs consistently better than existing rounding strategies, and it can reduce the error by additional 10 to 14 percent.

Other distortion measures for parameterizations may be highly application dependent, and one can argue whether misalignment, stretching, or bending of parameter lines is worse. The rounding error E_H, however, introduces random disturbance to the parameterization and one can simply say: lower *is* better, independently of the application. Because the rounding error reduction comes at low computational cost, the optimization is very worthwhile.

From the perspective of branch point balancing, lower rounding errors also mean that branch points can be added at a lower distortion penalty, and so, also the curl distortion can be further reduced.

The Right Number of Branch Points

Many authors of parameterization algorithms have tried to avoid high numbers of branch points instead of using their ability to positively influence the parameterization. By developing an effective method to create distortion-minimizing branch points, we demonstrated—independently of Myles and Zorin—that distortion can be significantly reduced by allowing new branch points, which often happen to form branch point chains.

With a precise estimation of the expected distortion introduced with each branch point, we do not run the risk of creating too many of them. We let our the algorithm decide which branch points should be kept to reduce the metric distortion, and which ones should be eliminated to avoid the rounding penalty with respect to the given parameter line density. A fundamental advantage of this approach is that the right number of branch points is automatically found for each line density without being dependent on any parameters. Many other methods (including our original QuadCover implementation) have only indirect control over the number of branch points, for example, by setting a smoothness parameters.

Speed

For medium-sized models of about 20 000 faces, our combination of methods works very well. For example, after 10 000 iterations, the distortion is reduced considerably and is within a few percent of what could be be achieved with more iterations. The whole parameterization and optimization process is then finished within about a minute. The numerics would be certainly faster if our code was implemented in C instead of Java, where optimized BLAS routines, faster solvers and hardware trigonometric functions are available.

While the cost of crucial computations grows just slightly worse than linear with the mesh size, the number of possible branch point positions grows much faster, and so our algorithms for branch point positioning and rounding need more iterations to complete. More complex models of around 100 000 triangles can take an hour to finish completely, because hundreds of branch points may have to be positioned for optimal results. This leads to a sheer endless number of possibilities to fine-tune the branch points and rounding. Thus, one has the choice of either fixing the solutions more greedily (and end up with sub-optimal results), or to take our route and try out a high number of choices to obtain a solution that is assured to be at least a local energy minimum. (So in essence, we blame the hardness of the problem for the rather slow convergence to a local minimum, although there may be more efficient algorithms than ours.) Nevertheless, large models, such as the bust example with 200 000 triangles in Figure 7.3 can be robustly parameterized, although in this case, the optimization took 100 CPU minutes to complete.

If faster results are required, the optimization processes can be stopped prematurely. The quality loss is then bearable, because most significant error reduction of either of our optimizations happens within the first few iterations.

Strengths and Limitations

Every algorithm has strength and weaknesses. QuadCover with its extensions excels on models of rather organic shape, which are not dominated by sharp features and other hard constraints. By the use of frame fields as a soft-constraint guidance, QuadCover parameterizations can adapt well to bends and curved large-area features. On these surfaces, branch points can be placed and moved rather freely, and thus are perfectly suited for our optimizations. If isometric parameterization is the goal, then—in our opinion—our extended QuadCover method places the branch points better than any other existing method.

As a drawback, parameter lines do not hit sharp edges exactly, as they do hard-constraint based methods like MIQ and the controlled distortion parameterization of Myles and Zorin. The missing feature has been added to QuadCover by my former colleague Matthias Nieser [Nie12], but more work is needed to make it work hand in hand with proper branch point placement and rounding optimization.

Our trial and error approach for branch point placement is not always efficient. If the parameter line density is very coarse, then it makes little sense to create hundreds of branch points if only a handful of them is kept in the end. However, with our approach could show what to expect from good branch point placement, and there might be interesting new ways to make it more efficient.

Concerning the efficiency of our method, there is also potential to simplify the work flow of field creation, field smoothing, branch point movement, integer rounding, and eventually the enforcement of hard constraints to fewer steps. With the combination of individual steps into one, the dependencies between them can be respected much better. Recent publications have achieved a lot of progress in this direction. On the other hand, the individual operations of the QuadCover pipeline could point out the clear distinction between curl distortion E_D and rounding distortion E_H and so enable the valuable analysis for branch point balancing.

Further Acknowledgments

All of our algorithms were implemented using our JavaView geometry processing software [PHK+], using CXSparseJ [Lin] as the only external software. All images of 3D models in this work are rendered using JavaView or POV-Ray, except those in Figures 6.2, 7.1, and 7.2, which contain images of the respective authors. The 3D models are courtesy of AIM@SHAPE, Stanford, and Max Planck Institute.

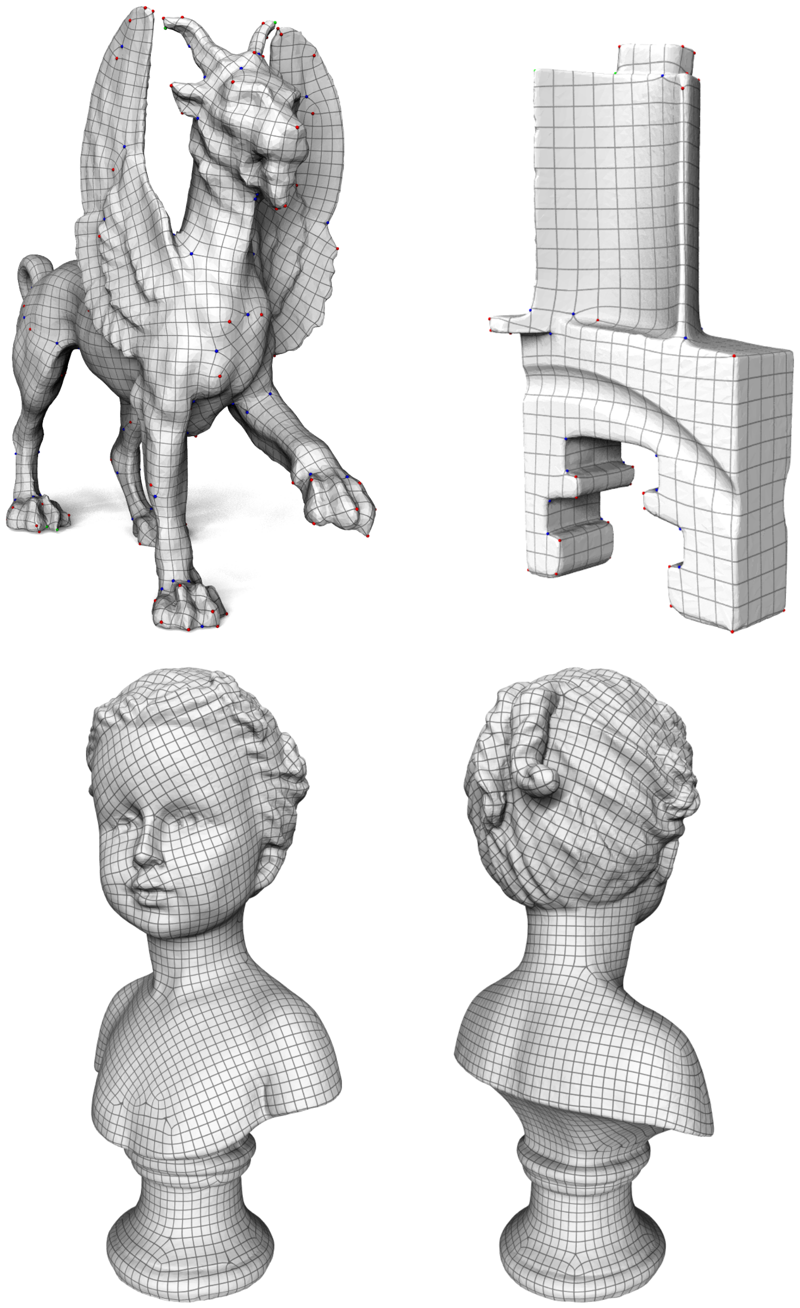


Figure 7.3: More examples of QuadCover parameterizations.

Bibliography

[ACSD$^+$03] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Trans. on Graphics*, pages 485–493, 2003.

[BCGB08] Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum*, volume 27, pages 449–458, 2008.

[BMRJ04] Ioana Boier-Martin, Holly Rushmeier, and Jingyi Jin. Parameterization of triangle meshes over quadrilateral domains. In *Eurographics Procedings/ACM Symposium on Geometry Processing*, pages 193–203, 2004.

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics*, 28(3):77:1–77:10, July 2009.

[CDHR08] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3):22:1–22:14, 2008.

[CSM03] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Procedings of the Symposium on Computational Geometry*, pages 312–321. ACM Press, 2003.

[Dav06] Timothy A. Davis. *Direct Methods for Sparse Linear Systems.* SIAM, 2006.

[DBG$^+$06] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM SIGGRAPH*, 2006.

[Den10] Linzhong Deng. Multiple-rank updates to matrix factorizations for nonlinear analysis and circuit design. PhD thesis, Stanford University, 2010.

[DKG05] Shen Dong, Scott Kircher, and Michael Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Design*, 22(4):392–423, 2005.

[dV10] Éric Colin de Verdière. Shortest cut graph of a surface with prescribed vertex set. In *Proceedings of the 18th annual European conference on Algorithms: Part II*, ESA'10, pages 100–111. Springer-Verlag, 2010.

[Dzi88] Gerhard Dziuk. Finite elements for the Beltrami operator on arbitrary surfaces. In Stefan Hildebrandt and Rolf Leis, editors, *Partial Differential Equations and Calculus of Variations*. Springer-Verlag, 1988.

[Ebe01] Wolfgang Ebeling. *Funktionentheorie, Differentialtopologie und Singularitäten*. Vieweg, 2001.

[EHP04] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.

[EW05] Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms*, pages 1038–1046, 2005.

[Gro99] Mikhail Gromov. *Metric structures for Riemannian and non-Riemannian spaces*, volume 152 of *Progress in Mathematics*. Springer-Verlag, 1999.

[GWY03] Xianfeng Gu, Yalin Wang, and Shing-Tung Yau. Computing conformal invariants: Period matrices. *Communications in Information and Systems*, 3(3):153–170, 2003.

[GY03] Xianfeng Gu and Shing-Tung Yau. Global conformal surface parameterization. In *Symposium on Geometry Processing*, pages 127–137, 2003.

[Haz93] Michiel Hazewinkel, editor. *Encyclopedia of Mathematics*, volume 9, chapter Two-Dimensional Manifold, pages 288–292. Kluwer Academic Publishers, 1993.

[Hod52] William V. D. Hodge. *The Theory & Applications of Harmonic Integrals*. Cambridge University Press, 1952.

[HP04] Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23(3):391–400, 2004.

[HZ00] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *ACM SIGGRAPH*, pages 517–526, 2000.

[HZM$^+$08] Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt, and Hujun Bao. Spectral quadrangulation with orientation and alignment control. In *ACM SIGGRAPH Asia papers*, pages 147:1–147:9. ACM, 2008.

[KNP07] Felix Kälberer, Matthias Nieser, and Konrad Polthier. QuadCover—Surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384, 2007.

[KNP10] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Stripe parameterization of tubular surfaces. In V. Pascucci, H. Hagen, J. Tierny, and X. Tricoche, editors, *Topological Methods in Data Analysis and Visualization*. Springer-Verlag, 2010.

[KP10] Felix Kälberer and Konrad Polthier. Frame field generation for mesh parameterization. *AIP Conference Proceedings*, 1281(1):1031–1034, 2010.

[KSS06] Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics*, 25(2), 2006.

[Lin] Richard W. Lincoln. *CXSparseJ: A Concise Sparse matrix package, Version 2.2.6*. Copyright 2006–2011, Timothy A. Davis, 2011–2012 Richard W. Lincoln. Available at `https://github.com/rwl/CXSparseJ`.

[LLL82] Arjen K. Lenstra, Willem H. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[LZX$^+$08] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. *Proceedings of the Eurographics Symposium on Geometry Processing*, 27(5):1495–1504, 2008.

[Mas64] William S. Massey. *Algebraic Topology: An Introduction.* Harbrace College Mathematics Series, 1964.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective.* Springer, 2002.

[Mil65] John Milnor. *Topology from the Differentiable Viewpoint.* Princeton University Press, 1965.

[MK04] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pages 207–216, 2004.

[MK06] Martin Marinov and Leif Kobbelt. A robust two-step procedure for quad-dominant remeshing. *Computer Graphics Forum*, 25(3):537–546, 2006.

[Mun84] James R. Munkres. *Elements of Algebraic Topology.* Addison Wesley, 1984.

[MZ12] Ashish Myles and Denis Zorin. Global parametrization by incremental flattening. *ACM Transactions on Graphics*, 31(4):109:1–109:11, July 2012.

[MZ13] Ashish Myles and Denis Zorin. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics*, 32(4):105:1–105:14, 2013.

[Nie12] Matthias Nieser. *Parameterization and Tiling of Polyhedral Surfaces.* PhD thesis, Freie Universität Berlin, 2012.

[NPPZ12] Matthias Nieser, Jonathan Palacios, Konrad Polthier, and Eugene Zhang. Hexagonal global parameterization of arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):865–878, 2012.

[PHK$^+$] Konrad Polthier, Klaus Hildebrandt, Felix Kälberer, Matthias Nieser, and Ulrich Reitebuch. JavaView — interactive 3D geometry and visualization software. Copyright 1999–2013. Available at `http://javaview.de`.

[Pię96] Artur Piękosz. Basic definitions and properties of topological branched coverings. *Topological Methods in Nonlinear Analysis, Journal of the Juliusz Schauder Center*, 1996.

[Pol02] Konrad Polthier. Polyhedral surfaces of constant mean curvature. Habilitation Treatise, Technische Universität Berlin, 2002.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[PP03] Konrad Polthier and Eike Preuss. Identifying vector field singularities using a discrete Hodge decomposition. In *Visualization and Mathematics III*, pages 113–134. Springer, 2003.

[PZ07] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Transactions on Graphics*, 26(3), July 2007.

[RLL$^+$06] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 25(4):1460–1485, 2006.

[RVLL08] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Transactions on Graphics*, 27(2):1–13, 2008.

[SSP08] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. In *ACM SIGGRAPH 2008 papers*, pages 77:1–77:11, 2008.

[Sti80] John Stillwell. Complex analysis and surface topology. In *Classical Topology and Combinatorial Group Theory*, volume 72 of *Graduate Texts in Mathematics*, pages 53–88. Springer, 1980.

[TACSD06] Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. Designing quadrangulations with discrete harmonic forms. In *Eurographics Symposium on Geometry Processing*, 2006.

[vTKP11] Christoph von Tycowicz, Felix Kälberer, and Konrad Polthier. Context-based coding of adaptive multiresolution meshes. In *Computer Graphics Forum*, volume 30, pages 2231–2245. Wiley Online Library, 2011.

[War06] Max Wardetzky. *Discrete Differential Operators on Polyhedral Surfaces—Convergence and Approximation.* PhD thesis, Freie Universität Berlin, 2006.

[WMKG07] Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun. Discrete Laplace operators: No free lunch. In *SGP '07: Eurographics Symposium on Geometry Processing*, pages 33–37, 2007.

[Zie98] Günter M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1998. 2nd edition.

List of Symbols

General Notes

To make the distinction between objects as clear as possible, we use different font styles for different types of mathematical objects whenever possible. In particular we use the following styles and marks:

bold face to denote vectors and matrices (exception: the stiffness matrix $\mathfrak{L}$), and simplices when interpreted as objects in $\mathbb{R}^n$ (e.g., the interior angle $\angle(\mathbf{e}, \mathbf{f})$)

sans serif to denote quantities (e.g., the number of vertices, v)

italic font is used for indices of geometric objects and vectors (e.g., $e = (s, t), \mathfrak{L}_{ij}$)

a tilde, $\widetilde{\ }$, to denote objects on coverings (e.g., the covering surface itself, $\widetilde{M}$)

an asterisk, *, for non-conforming objects (e.g., non-conforming functions, S_h^*)

a bar, $\bar{\ }$, to denote objects in complex notation (e.g., a complex vector $\bar{\mathbf{u}}$)

List of Frequently Used Symbols

M	A (geometric) simplicial surface or triangle mesh	12
$\mathfrak{K}$	An abstract simplicial complex	12
$\sim$	Adjacency: $\sigma \sim \tau \Leftrightarrow \sigma$ is adjacent to τ	12
$\mathsf{V}, \mathsf{E}, \mathsf{F}$	Set of vertices, edges and faces of a (geometric) simplicial complex	13
$\mathsf{v}, \mathsf{e}, \mathsf{f}$	Number of vertices, edges and faces of a simplicial complex	13
g	Genus of a simplicial surface	13
$\mathbf{A}^\mathsf{T}$, $\mathbf{A}^\mathsf{H}$	Transpose and conjugate transpose of a matrix or vector	17, 72
S_h	Space of continuous PL functions (hat functions)	17
S_h^*	Space of PL functions with edge-midpoint continuity	18
φ	Lagrange basis functions, basis functions of S_h	18
ψ	Crouzeix-Raviart basis functions, basis functions of S_h^*	18

$\mathbf{J}$	Rotation by $\frac{\pi}{2}$ in the tangent plane	18
$A_{\mathbf{t}}$, $A_{\mathbf{e}}$	Area of triangle t, area share of edge $e = (s,t)$, $A_{\mathbf{e}} = \frac{1}{3}(A_{\mathbf{s}} + A_{\mathbf{t}})$	18
∇	Gradient of a function	18
$E_{\mathrm{D}}(u)$	Dirichlet energy of a piecewise linear function $u \in S_h^*$	19
$\mathfrak{L}$, $\mathfrak{L}^*$	Conforming and nonconforming stiffness matrix	19, 20
Δ	Discrete Laplace-Beltrami operator	20
L	Cholesky factor of $\mathfrak{L}$ or $\mathfrak{L}^*$	25, 72
$\mathcal{C}$	Vertex-edge incidence matrix	21
$\mathfrak{X}$	Space of piecewise constant tangential vector fields	23
L^2, $\|\|X\|\|$	Norm of a vector field X, $\|\|X\|\| = \left(\int_M \|\|X_t\|\|^2_{\mathbb{R}^3} \mathrm{d}A\right)^{\frac{1}{2}}$	23, 70
$\mathcal{P}, \mathcal{C}, \mathcal{H}$	Space of conforming potential, non-conforming co-potential and harmonic vector fields	23
$\mathcal{P}^*, \mathcal{C}^*, \mathcal{H}^*$	Space of non-conforming potential, conforming co-potential and harmonic vector fields (alternative discretization)	23
$X_{\mathcal{P}}, X_{\mathcal{C}}, \ldots$	Projection of X to $\mathcal{P}, \mathcal{C}, \ldots$, *i. e.*, the potential part, curl part, etc. of X	24
Σ_k	Locally supported vector field with period 1 w. r. t. path γ_k	29, 59
$H_{\mathbf{b}}$	Harmonic vector field on M with periods $\mathbf{b}$ (w. r. t. $\gamma_0, \ldots, \gamma_{2\mathsf{g}}$)	30
$\mathbf{b}_k(X)$	Period of a vector field X along a curve γ_k of the homotopy basis	29, 41
$\mathbf{d}_k(X)$	Integer offset of $\mathbf{b}_k(X)$, $\mathbf{b}_k(X) - \mathbf{d}_k(X) \in \mathbb{Z}$	41
θ_e, θ_{st}	Curvature of a vector field or frame field at edge $e = (s,t)$	33
m_e, m_{st}	Matching of a vector field or frame field at edge $e = (s,t)$	44
R	Frame switching operator, conformal structure on frame fields	44
$\mathfrak{F}$	Space of piecewise constant tangential frame fields	45
$(X,Y)^{\perp}$	Orthogonalized frame	46
$\widetilde{M}$	Covering surface of M with respect to a given matching	48
π	Covering map $\pi : \widetilde{M} \to M$	48
$\mathsf{b}_{\mathrm{even}}$, $\mathsf{b}_{\mathrm{odd}}$	Number of branch points with even and odd matching index	51
$\tilde{\mathsf{h}}_{\mathrm{sym}}$	Number of independent symmetric harmonic fields on the covering	58
$E_D(X)$	Amount of curl distortion, $E_D(X) = \frac{1}{2}\|\|X_{\mathcal{C}}\|\|^2$	71
E_H	Size of harmonic correction field, $E_H = \frac{1}{2}\|\|H_{\mathbf{d}}\|\|^2$	71
Δ_{E_H}	Expected reduction of $E_H(X)$ per cancelled branch point pair.	76, 84
$\Delta_{E_D}(p,q)$	Change of E_D when canceling two branch points p and q	76, 85
$\mathbf{P}$	Period Matrix	79
r	Recursion depth in the rounding optimization: r entries of $\mathbf{c}$ are changed at a time.	80

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Parametrisierung simplizialer Flächen. Darunter versteht man das Erzeugen einer Abbildung zwischen einer Fläche und der euklidische Ebene, um durch diese Korrespondenz die vorhandene Struktur der Ebene auf der Fläche nutzbar zu machen. Zum Beispiel besitzt die Ebene eine natürliche Rasterung in Einheitsquadrate, die mithilfe der Parametrisierungsfunktion auf die Fläche übertragen werden kann. Anwendungen hierfür sind zum Beispiel die Neuvernetzung und Texturierung von Flächen, und die Erstellung von Kontrollnetzen zur Generierung von Subdivisions- oder NURBS-Flächen.

Parametrisierungsfunktionen haben meist eine Reihe von Gütekriterien zu erfüllen, wichtig ist zum Beispiel geringe Längen- und Winkelverzerrung. Oft ist zusätzlich gefordert, dass die Gradienten der Abbildung mit der Ausrichtung von Flächenmerkmalen – etwa von scharfen Kanten – übereinstimmen.

Unser QuadCover-Verfahren, das die Grundlage dieser Arbeit bildet, erzeugt automatisch aus einem Tensorfeld von Merkmalsrichtungen eine Parametrisierung. Das Verfahren basiert auf der Grundlage, dass diese mehrdimensionalen Tensorfelder als eindimensionale Vektorfelder auf einer verzweigten Überlagerung der Fläche interpretiert werden können. Auf diese Weise können bekannte Resultate über Vektorfelder, zum Beispiel die Hodge-Zerlegung, angewendet werden. Auf dieser Basis findet QuadCover die Parametrisierung, die einem gegebenen Richtungsfeld am nächsten kommt.

Für Parametrisierungen höchster Güte muss zusätzlich die Längen- und Winkelverzerrung minimiert werden. Hierfür ist die Anzahl und Position von Verzweigungspunkten im Richtungsfeld entscheidend. In dieser Arbeit setzen wir an drei unterschiedlichen Punkten an: Erstens, zeigen wir mit einem neuen Verfahren, dass die Verzerrung durch das Verschieben und vor allem durch das Erschaffen von Verzweigungspunkten drastisch minimiert werden kann. Zweitens wird die Verzerrung, die durch die Existenz von Verzweigungspunkten entsteht, durch ein neues Rundungsverfahren deutlich stärker verringert als mit bisherigen Methoden. Der dritte Ansatz stellt die unterschiedlichen Arten von Verzerrung der zuvor genannten Verfahren gegenüber, so dass daraus die optimale Anzahl von Verzweigungspunkten bestimmt werden kann. Die Kombination der Ansätze erlaubt es, auch neue Verfahren hinsichtlich der Verzerrung zu übertreffen.